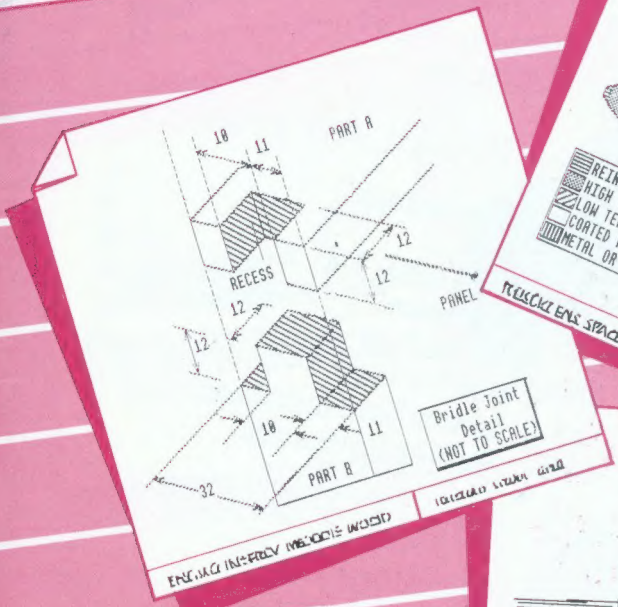
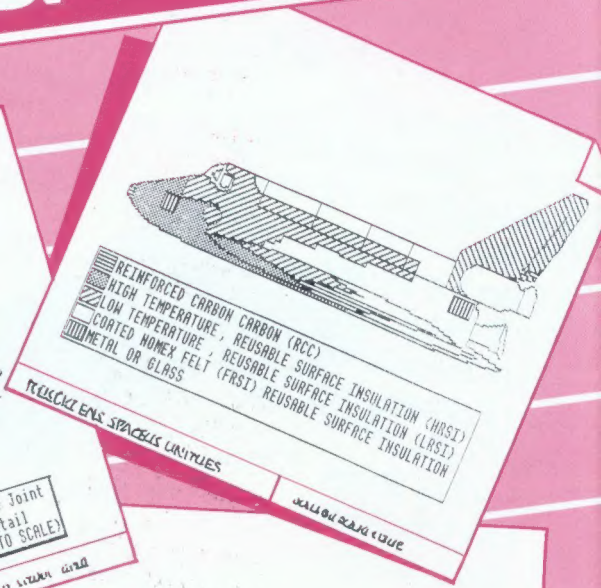
Vol.7 No.5 October 1988

BEEBUG

FOR THE BBC MICRO & MASTER SERIES





ASTAAD 3 Computer Aided Design

- PRINT FORMATTING
- DESIGNING ICONS
- INTEGRA-B REVIEW
- USING A MOUSE

BEEBUG Vol.7 No.5 October 1988

FEATURES

REVIEWS

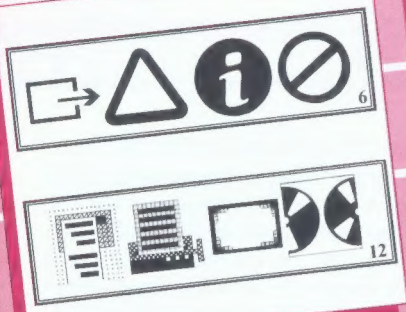
REGULAR ITEMS

HINTS & TIPS

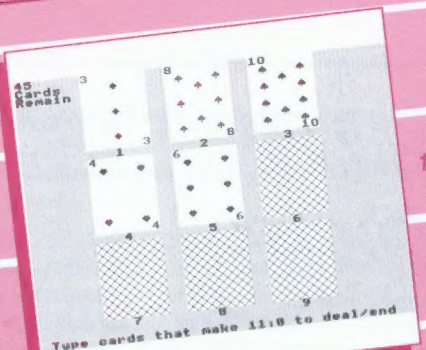
PROGRAM INFORMATION

All programs listed in BEEBUG magazine are produced direct from working programs. They are listed in LISTO1 format with a line length of 40. However, you do not need to enter the space after the line number when typing in programs, as this is only included to aid readability. The line length of 40 will help in checking programs listed on a 40 column screen.

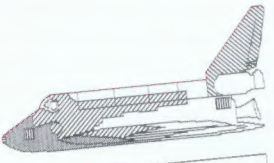
Programs are checked against all standard Acorn systems (model B, B+, Master, Compact and Electron; Basic I and Basic II; ADFS, DFS and Cassette filing systems; and the Tube). We hope that the classification symbols for programs, and also reviews, will clarify matters with regard to compatibility. The complete set of icons is given



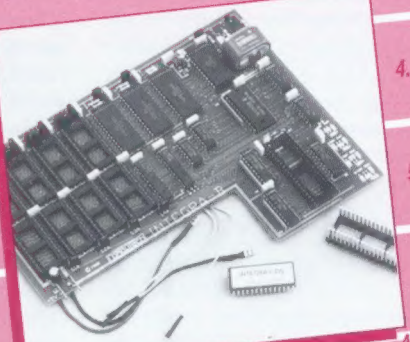
1.



2.



3.



4.

#####.##	#####.##	#####.##
1.00	-3.14	9.87
+#####.##	+#####.##	+#####.##
+1.00	-3.14	+9.87
00000.##	00000.##	00000.##
00001.00	-0003.14	00009.87
+0000.##	+0000.##	+0000.##
+0001.00	-0003.14	+0009.87

5.

6.

below. These show clearly the valid combinations of machine (version of Basic) and filing system for each item, and Tube compatibility. A single line through a symbol indicates partial working (normally just a few changes will be needed); a cross shows total incompatibility. Reviews do not distinguish between Basic I and II.

Computer System		Filing System	
Master (Basic IV)	M	ADFS	A
Compact (Basic VI)	C	DFS	D
Model B (Basic II)	II	Cassette	
Model B (Basic I)	I	**Tube Compatibility**	
Electron	E	Tube	

Editor's Jottings

MICRO USER SHOW

The autumn Micro User Show will be held at the New Horticultural Hall, Westminster from 11th-13th November. BEEBUG will have a prominent stand at this show with BEEBUG and RISC User magazines and discs. We shall also be featuring our own range of software and hardware products for the BBC and Archimedes computers, including the internal modem for the Master 128, and Hearsay, our comms software for the Archimedes. Not only that but Acorn's amazing new operating system for the Archimedes, RISC OS, will be demonstrated on our stand. We hope that as many members as possible will visit us at the show. We would be delighted to meet you and talk with you. The Micro User ad in this issue entitles you to £1 off entry for tickets ordered in advance.

MICRONET

There have been many changes to the BEEBUG pages on Micronet recently, with new page logos, changed routing, and the revamping of large parts of our database. Check the BEEBUG pages for the latest reviews and information, including coverage of the Archimedes. BEEBUG is also implementing a new policy for telesoftware which should see a completely revised and upgraded programme of software appearing during the next few months. Watch out too for our amazing free telesoftware bonus for the forthcoming Christmas season.

In order that telesoftware may be downloaded by BEEBUG members without incurring page charges, we use a password system. So that both BEEBUG and RISC User members may have equal access, we shall be publishing the password at the foot of the editorial page in each issue of both magazines starting with this issue. The information on Micronet will tell you which magazine's password is required for any particular program.

BEST OF BEEBUG

We have published many excellent programs in BEEBUG in recent years. Although these are available on the monthly magazine discs we have decided to group the best of these programs together under a number of thematic titles. In this way you can obtain all the best programs on a particular subject for the price of just a single disc. Full details of our first two releases are contained in the supplement. There has never been a better nor a cheaper way of buying the best of BEEBUG programs.

This month's telesoftware password is ***blackbird.***

News News News Ne

RISC OS MADE PUBLIC

Acorn has finally put its Arthur 2 operating system for the Archimedes on show to the public. The new operating system, which Acorn are calling *RISC OS*, offers many improvements over the current Arthur 1.20. The main changes are the ability to run several programs simultaneously, and a much improved Desktop system, which now allows files to be moved around just by dragging them between windows, very much like the Apple Macintosh. There are also many other new features, including a RAM filing system, and a complete set of object based drawing commands, which makes the implementation of drawing packages very simple. *RISC OS* will not be available until April next year, although Acorn say this is due to the lead time for producing ROMs, and the software is in fact complete. The documentation and the three disc Welcome suite, which will contain several useful applications, have yet to be finalised. The cost of upgrading to *RISC OS* is expected to be around £50.

ACORN BACK IN THE BLACK

Acorn has announced a profit of £1.05 million in the first six months of this year. This is a great improvement on the £0.95 million loss for the same period last year. The earning per share is 1.1p. Acorn's chairman, Elserino Piol, said that the turnaround in fortune is due to both the company's concentration on mainstream products, and increasing sales of the Archimedes. Hopefully, this improvement in Acorn's finances will increase the faith that others have in the company.

PERSONAL COMPUTER SHOW

For the first time in several years, Acorn made an impact at the Personal Computer show (formerly the PCW show). In common with last year, Acorn not only displayed its own products, but also accommodated some of the many third party suppliers for Acorn machines. Acorn's major exhibit was RISC OS (see above), but it was also showing its new Desktop Publishing (DTP) package, which is based on the Timeworks system from GST Ltd. *Acorn DTP* supports all the functions you would expect from a page-layout package, including the ability to import graphics and

text. This, combined with the multi-tasking of RISC OS allows for a very powerful setup. The only real limitation of *Acorn DTP* is the lack of different text fonts, and it is hoped that this is something which will quickly be rectified.

Clares Micro Supplies was showing its new *Pro-Artisan* drawing package for the Archimedes. The original Artisan package, which was one of the first releases for the Archimedes, has been highly praised, and it is on this that *Pro-Artisan* is based. Unlike the original, *Pro-Artisan* works in the 256 colour mode 15, with the colours being chosen using simple colour charts. A completely new feature is the ability to distort an image, in the form of a sprite, around the surface of a 3D object. The colour fill has been extended to allow a graduation of shades, which is very useful when rendering 3D shapes. A novel feature of *Pro-Artisan* is the wash facility. If the airbrush is used to spray an area, washing that area will have the effect of splashing a drop of water on the paper, making the image blur outwards. *Pro-Artisan* sells for £165 inc. VAT, which is rather more than the original Artisan. More details from Clares Micro Supplies, 98 Middlewich Road, Northwich, Cheshire CW9 7DA, tel. (0606) 48511.

PUNCH 'EM UP

Superior Software has just released a new boxing game for the Beeb and Electron. The game, called *By Fair Means or Foul*, includes all the facilities you would expect of a boxing simulation, and also allows you to cheat by using head butts, groin punches etc, but only when the ref is not looking. *By Fair Means or Foul* on disc for the model B and Master costs £11.95. The tape version costs £9.95, as does the Electron version. For a Master Compact the disc is £14.95. Superior Software, Regent House, Skinner Lane, Leeds LS7 1AX, tel. (0532) 459453.

SNATCH A PICTURE

Snatch from 4Mation is a screen grabber and dumper for any BBC micro with sideways RAM. *Snatch* allows the screen to be grabbed while a program is running, and then dumped to either an Epson compatible printer, or an Integrex 132 in full colour. The colours can be modified before printing, and there is a choice of sizes. *Snatch* costs £18.40 inc. VAT from 4Mation, Linden Lea, Rock Park, Barnstaple, Devon EX32 9AQ, tel. (0271) 45566.

MATHS IS FUN WITH LOGOTRON

The educational software house Logotron, which is most famous for its implementation of the Logo programming language, is about to launch a package to help primary school children learn maths. *Numerator*, as the new package is called, allows mathematical problems to be modelled around three types of component: 'Tanks' which hold numbers, 'Boxes' which perform numeric operations, and 'Pipes' which connect Tanks to Boxes. Input can be taken from the Beeb's analogue port, and the results can be plotted out. *Numerator* costs £39.10 inc. VAT, with an Econet site licence available for a further £200. Logotron can be found at Dales Brewery, Gwydir Street, Cambridge CB1 2LJ, tel. (0223) 323656.

NIDD VALLEY PRICES

Unfortunately, the prices given in the review of *Illustrator* and *Paintbox* (now renamed *Colourbox*) from Nidd Valley Micro Products in the last issue of BEEBUG were not correct. The correct prices are given below. All of these include VAT and postage, and the prices in brackets include the *Digimouse*.

Paintbox	£14.95 (£44.90)
Illustrator	£19.90 (£49.90)
Paintbox & Illustrator	£29.90 (£59.90)

Additionally, until the 31st of January, Nidd Valley are offering special winter prices. *Paintbox* and *Illustrator* are available together with *Chauffeur* and *Grafik* for just £29.90, or £49.90 with a *Digimouse*. *Chauffeur* is a package to allow the mouse be used with programs that only support keyboard input, and *Grafik* is a simple drawing package. The *Digimouse* on its own with no software is available for £29.90. Nidd Valley Micro Products are at 4AA9 Thorp Arch Trading Estate, Wetherby, West Yorkshire LS23 7BJ, tel. (0937) 844661.

ICON DESIGN

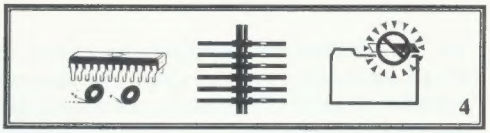
Graphic designer Roger Burg offers some expert advice on the design of icons to help you make the most of the BEEBUG Icon Designer.

I'm still surprised at how many icons surround us every day. Yet in computer software they often present a barrier rather than an aid to understanding. This really shouldn't be so. Conveying meaning through images is not always easy, but on the computer, icons are used not so much to explain new subjects, but to extend the user's current understanding. For this reason they are less vulnerable to jargon and imprecision than computer terms like 'file', 'bank' and 'cell', which have many different dictionary meanings without considering their technical interpretation. In this article we'll take a look at a wide range of icon designs (the references relate to the illustrations).

The icon for 'disc write-protected' shows the cause of the failed save operation as well as its solution, unlike the prompt where the jargon sends the novice back to the manual (1).

The common word processing icons convey format options far more clearly than diverse technical terms (2).

Icons can incorporate text, or present optional verbal prompts, when language is not a barrier (3).



Images - like any other means of communication - fail when they require a big jump in understanding, or when the user isn't really familiar with the concept, as with 'relocatable module', 'hard disc', and 'file locked' (4).

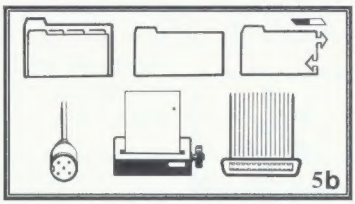
But icons, like words, take their meaning from context, and the order in which they appear is an important part of their design. If the user selects a text *style* option, then tilted, italic, extended and bold make sense. If he's selecting printer output, then the parallel and serial options need no more explanation. Related ideas, like files and directories lend meaning to each other (5a and 5b).

Some abstract ideas have widely established symbols, from arrows to hearts. Particularly useful are the international exit symbol, warning triangle, information, and the negation bar (6).

The exit symbol is a good challenge for the aspiring artist. There are dozens of shapes for an arrow head, and some 'move' better than others. A good designer will balance the white-space surrounding the box and arrow to convey the sense of emerging for those unfamiliar with the sign.

Many other international symbols and common images can be adapted in screen menus, though

it is treacherously easy to crib a good symbol, only to find that it has a precise definition quite different from the one intended.



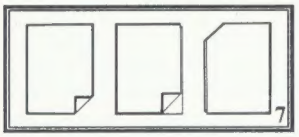
different, which can wreak havoc with screen layout.

Linguistic and cultural conflicts can be a minefield. Puns like the ram for 'RAM', or the bee for 'busy' are easy to check for. Innocent assumptions are not. Arabic cultures generally use the 'indian', not our 'arabic' numerals (9).

Ticks and crosses reverse their meanings on the other side of the Atlantic because American teachers put a 'check' (or tick) on answers which the pupil should check. Crosses are reserved for right answers. But some European software writers still use ticks and crosses to indicate whether a menu option has been selected or not.

PLANNING AHEAD

Before committing yourself to an icon, it's important to sketch out the images which may be needed in the foreseeable future. If you use the convention of the folded corner to identify a sheet of paper, you won't be able to use it later to show other turnable pages of 'help' (7).



Mail-related images present similar problems. In England, mail boxes (for receiving mail) are uncommon, and we tend to confuse them with the pillar box, for sending it. But if you've used the bulging pillar box to show that electronic mail has arrived, then it can't be used later, when you want to show that the system still has mail to send. The sealed envelope is widely used to indicate mail arrived, but if you prettify it by adding the stamp and address, your icons for mailing lists and address labelling, if you add them, will be less clear (8).

If a smile is part of the world's universal language, the good old smiley and the frown convey right and wrong better, and illuminated switches represent on and off. Running and walking are clearer than the hare and tortoise to indicate speeds, and the face and body generally convey many ideas common to all races (10).

But whatever our cultural and linguistic differences, anyone using a micro is likely to recognise stationery, machinery and electrical equipment, and the micro itself is often a valuable source of images (11).

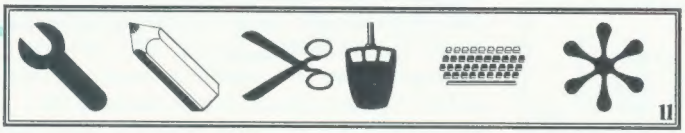
COMMERCIAL VALUE

Icons can reduce the cost of modifying software for different countries. Textual prompts must be translated, and even when they print from left to right and share a similar character set, the lengths of the messages will be

SOFTWARE FACTORS

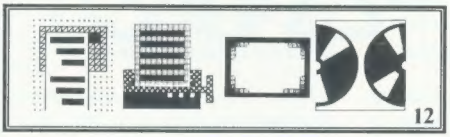
The micro puts its own limitations on what can

be designed. The BBC was built to cope with fuzzy televisions, and to compress 80-column characters into half width. So BBC icons like its



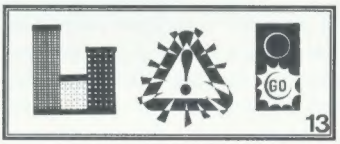
character set must generally avoid using thin verticals and diagonals to remain legible and to present a coherent style.

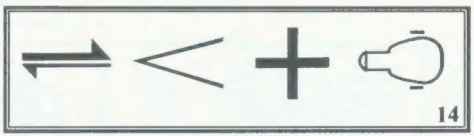
The restrictions of memory and screen space are extreme, but several traditional graphic practices can help. Coding in shapes rather than outlines saves pixels and the simplicity is visually more effective. Sometimes adjacent blocks can be distinguished by colour. Finally, if an image is still too big, slice a third or even two-thirds away - simplification is effective if the rest of the image is clearly implied (12).



But the micro supplies some solutions too. As tonal value is more assertive than colour, it often helps to use colour to adjust the weight and direction of a line, or profile. Acorn machines in particular can store and print large images very economically as strings of VDU characters.

And the computer offers its own opportunities. When the pointer touches them, icons can move. Verbs like 'compress', 'move', 'rotate', 'append', 'extend' and 'unplug' cry out for simple animation. Similarly, a palette, or a stop sign seems to require colour, and warnings convey more urgency in flashing colour (though coloured images should still be understandable on monochrome screens) (13).





SOURCES

The best icons are not stunning. They are the ones which, in use, seem so obvious and natural as to defy comment. So they are often difficult to recall when you need them.

Probably the most reliable way to find a new icon for a well-known idea is to flick through books in that subject area, in the local library. Diagrams, pictures and cover designs usually reveal someone else's well considered images for the subject.

Original icons for new ideas are more difficult, and collecting ideas is perhaps the most valuable part of the exercise. Inspiration is important, but unreliable! Books on modern graphic design and traditional symbols are helpful, but for fun as well as convenience I keep notebooks of images culled from computer and graphics magazines, public signs, dash-board buttons, hi-fi, video and TV controls, TV news graphics and weather charts, trade marks and logos, washing and cooking instructions and countless everyday sources.

A few reference books can be invaluable, like the Highway Code, rub-down lettering catalogues and signmaker's brochures. British standards and many specialist associations list signs suitable in their own areas, and books on contemporary specialisms use their own symbolic codes, like chemistry, music, maths, theatre, circuitry and many others (14). I hope you'll find these pages worth keeping too.

Note: the BEEBUG Icon Designer was published in Vol.7 No.2 as part of our Mini Wimp series.

B

INTEGRA-B

Bernard Hill plugs in the latest piece of hardware that claims to turn your model B into a Master.

Product	Integra-B
Supplier	Computech The Garth, Hampsfell Road, Grange Over Sands, Cumbria LA11 6BG. Tel. (0448) 44604
Price	£122.50 inc. VAT

In BEEBUG Vol.6 No.9 I reviewed the Master Emulation ROM from Dabs Press: an inexpensive attempt to turn your Beeb into a Master. Now we have a different approach from Computech in the form of a hardware and software package: the Integra-B board.

Because of its hardware content, this is a much more complete solution than that from Dabs Press, and therefore inevitably in quite a different price league. In its basic form it consists of a ROM expansion board of about 8" by 6", which fits in the top-left corner of your Beeb. Some self-adhesive plastic strips are provided which support the board in this position, the opposite corner being supported by a 40-pin header plug which fits into the 6502 processor socket. On the Integra board is a socket into which the CPU chip should then be re-located. Integra-B does not provide the extended 65C02 processor of the Master.

Although the board fits directly over the Econet section of the circuit board, it can still be fitted with this in place, but requires a tiny modification to one support strip (details from Computech). Even though the board partly overlaps the motherboard's DFS section, it still has at least 1/4" clearance over an Acorn 1770 DFS card. The Integra board itself is finished to a particularly high standard of construction and installation takes only a few minutes, with the removal and relocation of two of the Beeb's power leads, one jumper and three flying leads to insert in the Beeb's PCB.

BOARD CONTENTS

Supplied on the board is a real-time clock, 32K of shadow RAM, 64K of sideways RAM and 8 empty sideways ROM sockets, together with the necessary support chips. Once the board is installed, the memory map is identical to that of a model B+, rather than to a Master.

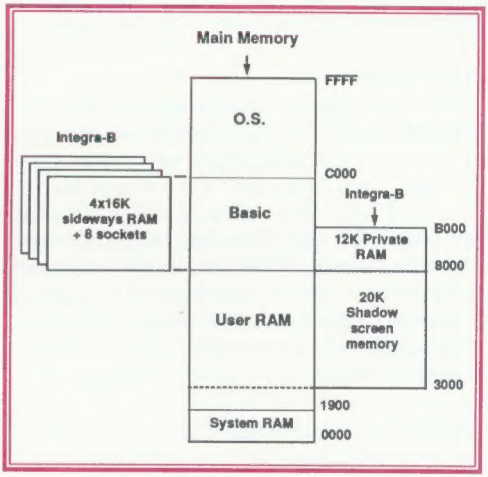


Figure 1. Integra-B Memory Map

The paged ROMS are allocated as follows:

0 - 3	BBC motherboard 4 x 16K ROM
4 - 7	Battery-backed 4 x 16K RAM
8 - 15	8 x 16K Integra-B sockets

The sideways RAM in slots 4 to 7 is fitted as two 32k RAM chips soldered into the Integra board, as is the 32k shadow RAM chip (see later).

CONFIGURATION OPTIONS

To emphasise the great degree of flexibility of this board, it contains no less than 19 jumpers: 16 of which allow configuration of the 8 sideways sockets on the Integra board. Thus within any pair of sockets 8-9, 10-11, 12-13 or 14-15, the following options are available:

A. Two 2764/27128 (8/16k) EPROMs (or ROMs)
B. One 27256 (32k) EPROM
C. One 62256 (32k) RAM chip (not supplied) which can be battery-backed if required.

The RAM in sockets 4-7 is permanently battery-backed, but removal of one of the board jumpers for a few seconds disables the supply to these chips and thus clears them. There are also two jumpers which allow write-protection of banks 4/5 and 6/7: write-protect switches are offered by Computech as an upgrade option. The functions of all the jumpers are printed on the board, which is a nice touch.

SOFTWARE

While the Integra-B board functions as a very well-behaved ROM/RAM expansion board all by itself, it needs supporting software to access the real-time clock and 32k of private and shadow RAM. This software comes as a 16k EPROM which should be fitted logically above any filing system ROMs: the manual suggests socket 3 on the BBC board, leaving the greatest flexibility to configure the Integra-B as RAM (you would need an extra four 32k RAM chips of course to make a full 12 sets of 16k RAM).

The facilities given by adding the control ROM can be divided into 5 sections as follows:

1. REAL-TIME CLOCK SUPPORT

Besides supporting a *TIME command, and the corresponding OSWORD 14 & 15 calls, the ROM has an astonishing range of *DATE routines which encompass on-screen calendar generation and day-of-the-week calculations. The plethora of formats allows you to perform such operations as finding the day of the week for a given date, finding the date of the next Sunday for example, finding the date of the first Monday in March, or even such things as finding the date of the third Sunday after 30 June next year, etc.

2. SHADOW AND PRIVATE RAM

The Integra-B ROM introduces an important new command, *OSMODE, which allows some measure of software reconfiguration of your model B. *OSMODE 0, specifies a normal model B, and Osmodes 1 to 4 are extensions to this. Each of these new "Osmodes" offers slightly different configurations of your Beeb, but all allow shadow RAM to be used. For example, after *OSMODE 1 screen modes 128 to 135 are available, as is a *SHADOW command, just as on the B+ or Master. The difference between Osmodes 1 to 4 is very badly described in the preliminary version of the manual. It turns out that the difference between the four Osmodes is the way in which various low-level operating system calls are implemented.

On the B+, the shadow memory takes 20k, and the remaining 12k of the 32k on board is rather awkward to access. Computech have encountered the same problem, but have used this memory area to give a number of additional facilities: besides saving some of the system parameters in this area, Computech has implemented an 8k printer buffer and a *BOOT command, which defines a sequence of operations to be performed on powerup (like *KEY10 on soft break). It appears that no attempt has been made to segment this 12k into different Master-type address ranges (ANDY and HAZEL), but the printer buffer concept has been extended to allow up to 4 sideways RAM banks to be allocated to a 64k buffer.

3. ROM SUPPORT

As might be expected from a product which attempts to imitate a Master, *ROMS, *INSERT, *UNPLUG, *SRLOAD and *SRSAVE are implemented, as is a new command, *SRWIPE, to clear a bank of sideways RAM. From version 1 of the Integra OS ROM (to be sent free to all owners of earlier versions), *SRDATA and *SRROM will also be available.

4. *CONFIGURE SUPPORT

*CONFIGURE options are provided as in the Master, the list being much the same, although LOUD, QUIET and (NO)SCROLL are missing, but a switch-on default Osmode

(see above) and an undocumented MOUSE option are added. Also included are commands to load and save the full set of configuration options via the current filing system, and a badly documented *CONFIGURE SHX ON/OFF option which, when enabled, ensures that as you switch, say, from mode 7 to mode 128, no programs are lost due to them being in the wrong memory bank.

5. OTHER * COMMANDS

The Integra OS ROM offers *APPEND, *SPOOLON, *STATUS and *PRINT as on the Master range. Also provided is *TUBE ON/OFF to control a second processor, and *BUFFER and *PURGE commands to control the printer buffer. Missing in comparison with the standard Master range, however are *CREATE, *GO(IO), *IGNORE, *SHOW, and *SHUT. An unusual *X*... command is provided to run a non-shadow utility from within a shadow memory session.

WORKSPACE

Those who remember my previous review of MER will recall that workspace was a particular problem. Even with all the extra RAM, Computech can never completely overcome this problem either, as some small part of the normal memory map will always be needed for sideways RAM access routines. But this has been limited on current versions of the software to just a few bytes in the CFS work area between &380 and &3A4. This means that Osmode 0 must be selected before writing to Cassette, although Computech inform me that they have plans for future versions to move the function key and character set buffers (pages &B and &C) to permanent expansions in private RAM and thus release 512 bytes for their own use.

THE MANUAL

The manual supplied with my board and EPROM (version 0.96) consists of 22 stapled sheets of A5, with an adequate coverage of installation and extra commands, but minimal explanation of board configuration, *OSMODE and other technical aspects. But again, users are promised a free replacement which should come with version 1 of the ROM.

COMPATIBILITY

Careful examination of this board and software enables one to conclude that the Integra-B system essentially puts some Master commands into a B+ memory map, and so therefore clearly cannot cope with Master-specific features such as ROM Polling, OSWRSC and (inevitably but sadly) filing system private workspace. Thus PAGE remains at &1900 and any Master programs needing a lower value may run out of space. However, Computech claim complete compatibility with all well-behaved software (such as the View family) and even with less polite software such as the Inter suite, Wordwise (Plus) and Genie, although sometimes only in Osmode 0.

THE FUTURE

The undocumented MOUSE configuration mentioned above, and and an as-yet incomplete page of the manual make it clear that Computech are currently working on a form of automated windows system making extensive use of sideways RAM. If this materialises then Integra-B will become a more powerful product, and much more than a simple expansion board.

CONCLUSION

So do you throw away your ATPL board, Solidisk Real-Time Clock and Aries Shadow RAM and then rush out and buy the Integra-B? If you have all these, then I suspect not, but if you are looking out for a shadow RAM system or extensive sideways RAM system, and possibly considering a real-time clock then maybe you should invest in this all-in-one solution. In terms of quality, design and facilities it cannot be faulted, and virtually converts your Model B into a B+ with some Master features (Real-Time Clock, system configuration and 64K sideways RAM). But full Master compatibility? Not quite. If that's what you want, you will probably be better off taking advantage of BEEBUG's deal to trade in your model B for a Master.

GRAPHIC DESIGN WITH ASTAAD

David Demaine presents a completely new and improved version of BEEBUG's highly acclaimed computer graphics program ASTAAD.

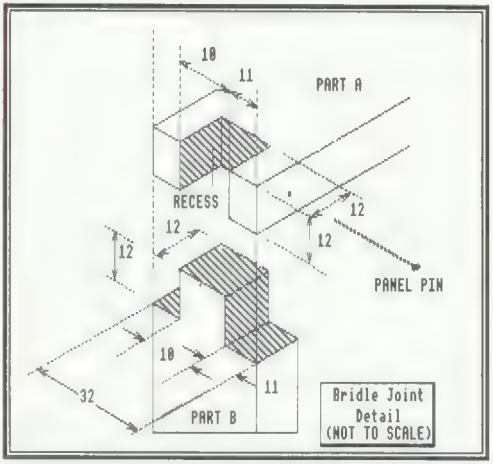
Tim Tonge's original ASTAAD program (published in BEEBUG Vol.2 No.7) was written to work on a standard model B and was, in all respects, as powerful as it could be for that machine. However, this new version is written specifically for the Master and makes use of both the additional memory and extra graphic facilities. The final product is a CAD program that is quite comparable, and in some ways more powerful, than the more popular commercial packages. For the model B owner we have put a copy of the original ASTAAD program on the magazine disc (and keystrips for both versions in the supplement).

Due to the length and complexity of the program it has been split into three separate articles. In this, the first article, we shall present the basic 'building blocks' for Astaad which will provide the simpler drawing functions. This first part has been written so that it is usable, even without the following enhancements. the next two articles we will expand upon these basic facilities and introduce advanced character generation, magnification, scaling, definable fill patterns, picture reversal, and much more.

RUNNING THE PROGRAM

Enter the program listing in the normal manner and save it before proceeding any further. Be careful to use the exact line numbers listed, make sure that you do not renumber it, or you will have difficulty appending the second and third parts.

When the program is run you will be presented with the ASTAAD drawing screen. As you might expect, you are given a single cursor in the middle of the drawing area. This cursor may be moved about using the conventional cursor keys. At the top of the screen your current co-ordinates will be displayed. Most of the functions available are called by using the red function keys.



We have provided a complete function key strip in this month's supplement. We would recommend that you cut it out and place it above the keyboard, allowing quick reference to a particular key's function. Because only a limited number of functions are available in this month's listing we have printed these functions in bold so that you will not get confused with the functions that are, and are not, initially available.

THE STATUS DISPLAY & SIMPLE FUNCTIONS

The top of the screen is used by ASTAAD for general information. As well as the current graphics co-ordinate, you will notice a great deal more information, including the current scale and the distance from the origin (called the vector). The current origin may be set anywhere on the screen by placing the cursor at the required position and pressing the Tab key. The rest of the display shows which functions

are currently active. The meaning of these will become apparent later.

The simplest functions are Move, Draw, and Rubout. Pressing any of these keys will alter the status display accordingly. Depending upon which of these are active, moving the cursor around the drawing area will simply move, rub out, or leave a line. Selecting the Line function will draw a line of any length in any direction. The Circle function will simply prompt for a radius and draw a circle around the current cursor position.

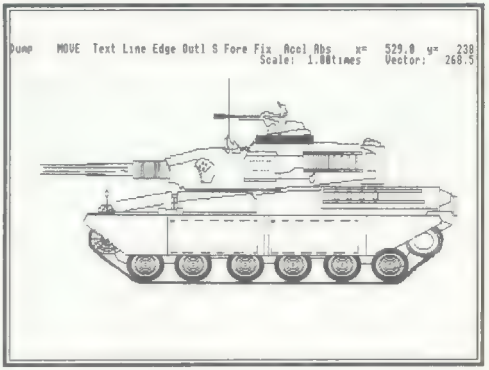
The Delete Area and Area Move functions make use of the vector position. To delete an area, move the cursor to the bottom left-hand corner of the area to be deleted and press the Tab key. Then proceed to the top right hand corner and select the Delete Area function. The area enclosed within these co-ordinates will be deleted. The Area Move function is used in much the same way, but the last two graphic origins will be recorded. Move to the bottom left-hand corner of the area, press Tab, then move to the top right hand corner and press Tab. Now move the cursor to the position to which the area is to be moved, select the Area Move function, and it will be moved.

The Infill/Outline will toggle the status display accordingly. Depending upon this, circles (and later other shapes) will be filled automatically when drawn. The colour reverse simply switches the foreground and background colours for future functions. The Save and Load Screen functions do exactly that. After having entered a filename, the current screen will be saved, or another screen will be loaded. Lastly, the Print Screen function will call a screen printing routine. The program, as listed, assumes that you have BEEBUG's DumpMaster ROM fitted. If this is not the case alter line 3270 to suit your particular software.

Like most CAD programs, the easiest way to learn how to use it is to play with it. As you can see from the screen shots included with this article, the package is very powerful and can be used to produce some extremely attractive artwork and diagrams.

```
   10 REM Program  Astaad (Part 1)
   20 REM Version  B3.02
   30 REM Author   David Demaine
   40 REM Based On Original By Tim Tonge
   50 REM BEEBUG   October 1988
   60 REM Program  Subject to copyright
   70 :
  100 ON ERROR MODE 3:PROCerror:END
  110 DIM OS% 40
  120 DIM ad(7),bd(7),cd(7),dd(7)
  130 MODE 128
  140 PROCinit
  150 PROCsetup
  160 :
  170 REPEAT
  180 PROCruler:PROCcursor
  190 key%=INKEY(0):J%=key%-128:IF key%<
=0 THEN UNTIL FALSE
  200 *FX 15,1
  210 IF accel% AND TIME<10 PROCaccel
  220 IF accel% AND TIME>=10 count%=0:s%
=4
  230 ON J%-12 PROCleft,PROCright,PROCdo
wn,PROCup ELSE 250
  240 UNTIL FALSE
  250 IF key%=127 MOVE x,y+28:PRINT CHR$
(127):x=x+16*(x>16):J%=14:UNTIL FALSE
  260 IF key%<127 AND key%>31 MOVE x,y+2
8:PRINT CHR$(key%):x=x-16*(x<(lx%-16)):J
%=13:PROCruler:UNTIL FALSE
  270 SOUND 17,-10,125,1
  280 IF key%=9 THEN curscode%=4:copycod
e%=5:linecode%=5:shapcode%=5:x3=x2:y3=y2
:x2=x1:y2=y1:x1=x:y1=y:curs$="MOVE":PROC
heading:J%=13:UNTIL FALSE
  290 IF key%=27 AND NOT INKEY-1 AND NOT
 INKEY-2 PROChome:UNTIL FALSE
  300 IF key%=27 AND INKEY-1 AND NOT INK
EY-2 CLG:PROCsetup:UNTIL FALSE
  310 ON ((J% DIV 16)+1) PROCfnkey,PROCs
hiftfnkey,PROCctrlfnkey,PROCshiftctrlfnk
ey ELSE UNTIL FALSE
  320 UNTIL FALSE
  330 :
 1000 DEF PROCsave
 1010 IF J%<>41 ENDPROC
 1020 fn$="Save  "
 1030 $OS%=FNcheck("File name?")
 1040 PROCmessage("Filename "+$OS%)
 1050 *FX 108,1
 1060 *SAVE NEERCS 3000 7FFF
 1070 *FX 108,0
 1080 OSCLI("RENAME NEERCS "+$OS%)
 1090 PROCrehead
```

```
1100 ENDPROC
1110 :
1120 DEF PROCload
1130 IF J%<>42 ENDPROC
1140 fn$="Load  "
1150 Fname$=FNcheck("File name?"):PROCm
essage("Filename "+Fname$)
1160 OSCLI("RENAME "+Fname$+" NEERCS")
1170 *FX 108,1
1180 *LOAD NEERCS 3000
1190 *FX 108,0
1200 OSCLI("RENAME NEERCS "+Fname$)
1210 ENDPROC
1220 :
1230 DEF PROCinit
1240 lx%=1280:ly%=960:VDU4:VDU 29|
1250 VDU 24,0;0;lx%-1;ly%-1;
1260 VDU 18,0,128,28,0,1,79,0
1270 VDU 5,23,1|
1280 VDU 19,0,3|19,1,4|
1290 *FX 4,2
1300 *FX 225,129
1310 *FX 226,145
1320 *FX 227,161
1330 *FX 228,177
1340 ENDPROC
1350 :
1360 DEF PROCsetup
1370 GCOL 0,128:GCOL 0,1
1380 x=lx%/2+1:y=ly%/2+2
1390 J%=13:lastJ%=4
1400 x1=x:y1=y:x2=x:y2=y:x3=x:y3=y:x4=0
:y4=0:vector=0
1410 s%=4:t%=50:count%=0:b$=STRING$(4,C
HR$32)
1420 fn$="Setup ":curs$="MOVE":com$="CA
D system initialised"
1430 shapcode%=5:curscode%=4:linecode%=
5:areacode%=0:copycode%=5
1440 linelen=400:lineang=PI/3:circrad=4
00
1450 polyrad=400:polysides%=7:polybase=
0:polyrot=PI/3:polytilt=PI/6
1460 soft%=0:soft$="Text":asize=16:ahei
ght=32:aspace=16:adotr=0.5:ainterp%=1:at
ext$="ASTAAD"
1470 copy%=FALSE:copy$="move"
1480 arrow%=FALSE:arrow$="Line  "
1490 circ%=TRUE:circ$="Circle"
1500 poly%=TRUE:poly$="Polygn"
1510 repeat%=FALSE
1520 fill%=0:fill$="Outl":ecf$="S":VDU
4,23,2,&FFFF;&FFFF;&FFFF;&FFFF;5
1530 col%=TRUE:col$="Fore"
```




```
1540 fix%=TRUE:fix$="Fix "
1550 accel%=TRUE:accel$="Accl"
1560 orig$="Abs "
1570 vector%=TRUE:vector$=" Vector:"
1580 dot%=0:dot$="Line"
1590 marg%=FALSE:marg$="Edge"
1600 scale$=STRING$(3,CHR$32):scl$="tim
es":scale=1.0:scalep=1.0
1610 PROCheading:PROCdelay:PROCrehead
1620 ENDPROC
1630 :
1640 DEF PROCheading
1650 VDU 4,12
1660 coma$=com$+STRING$(41-LEN(com$),CH
R$(32))
1670 @%=&00020108:PRINTTAB(0,0)fn$" "cu
rs$" "soft$" "copy$" "dot$" "marg$" "fil
l$" "ecf$" "col$" "fix$" "accel$" "orig$
" x=" (x-x4)*scale"y="(y-y4)*scale;
1680 @%=&00020206:PRINTTAB(0,1)coma$;CH
R$32;"Scale:"scalep;
1690 @%=&00020108:PRINT scl$ scale$ vec
tor$ vector*scale;
1700 VDU 5:ENDPROC
1710 :
1720 DEF PROCrehead
1730 com$="":fn$="Cursor":PROCheading
1740 ENDPROC
1750 :
1760 DEF PROCmessage(c$)
1770 com$=c$
1780 PROCheading
1790 ENDPROC
1800 :
1810 DEF FNinput(c$)
1820 PROCmessage(c$):VDU 4
1830 INPUT TAB(LEN(c$)+1,1)p:VDU 5:=p
1840 :
```

```
 1850 DEF FNcheck(c$)
 1860 PROCmessage(c$):VDU 4
 1870 INPUT TAB(LEN(c$)+1,1)p$:VDU 5:=p$
 1880 :
 1890 DEF PROCaccel
 1900 count%=count%+1
 1910 IF count%<61 s%=4+(count%*count% D
IV 256)*4 ELSE s%=256
 1920 TIME=0:ENDPROC
 1930 :
 1940 DEF PROCleft:x=x-s%*((x>=s%) OR ma
rg%)/((s%=4)-1)+(x-1)*((x<s%) AND NOT ma
rg%):PLOT curscode%,x,y:copycode%=5:ENDP
ROC
 1950 :
 1960 DEF PROCright:x=x+s%*((x<lx%-s%) O
R marg%)/((s%=4)-1)-(lx%-x-1)*((x>=lx%-s
%) AND NOT marg%):PLOT curscode%,x,y:cop
ycode%=5:ENDPROC
 1970 :
 1980 DEF PROCdown:y=y+s%*((y>=s%) OR ma
rg%)+(y-2)*((y<s%) AND NOT marg%):PLOT c
urscode%,x,y:copycode%=5:ENDPROC
 1990 :
 2000 DEF PROCup:y=y-s%*((y<ly%-s%) OR m
arg%)-(ly%-y-2)*((y>=ly%-s%) AND NOT mar
g%):PLOT curscode%,x,y:copycode%=5:ENDPR
OC
 2010 :
 2020 DEF PROCfnkey
 2030 ON J% PROCnoproc,PROCareamove,PROC
liner,PROCcirc,PROCnoproc,PROCnoproc,PRO
Cmove,PROCdraw,PROCrubout,PROCdelete,PRO
Cnone,PROCcopy ELSE ENDPROC
 2040 ENDPROC
 2050 :
 2060 DEF PROCshiftfnkey
 2070 ON J%-16 PROCnoproc,PROCnoproc,PRO
Cnoproc,PROCnoproc,PROCnoproc,PROCfill,P
ROCcolour,PROCnoproc,PROCnoproc,PROCnopr
oc ELSE ENDPROC
 2080 ENDPROC
 2090 :
 2100 DEF PROCctrlfnkey
 2110 ON J%-32 PROCdump,PROCnoproc,PROCn
oproc,PROCnoproc,PROCnoproc,PROCnoproc,P
ROCnoproc,PROCnoproc,PROCsave,PROCload E
LSE ENDPROC
 2120 ENDPROC
 2130 :
 2140 DEF PROCcursor
 2150 FOR I%=0 TO 1
 2160 *FX 19
 2170 PLOT 4,x,y+15:PLOT 6,x,y-15:PLOT 4
,x-15,y:PLOT 6,x+15,y
 2180 NEXT I%
 2190 PLOT 4,x,y
 2200 ENDPROC
 2210 :
 2220 DEF PROChome
 2230 IF x4=0 AND y4=0 THEN x=lx%/2+1:y=
ly%/2+2 ELSE x=x4:y=y4
 2240 J%=13
 2250 ENDPROC
 2260 :
 2270 DEF PROCruler
 2280 IF J%<13 OR J%>18 ENDPROC
 2290 IF J%=17 ENDPROC
 2300 VDU 4
 2310 X=x1-x:Y=y1-y
 2320 vector=SQR(X*X+Y*Y)
 2330 @%=&00020108:PRINTTAB(61,0)(x-x4)*
scale TAB(72,0)(y-y4)*scale;
 2340 PRINTTAB(70,1) vector*scale;
 2350 TIME=0
 2360 VDU 5:ENDPROC
 2370 :
 2380 DEF PROCareamove
 2390 IF J%<>2 ENDPROC
 2400 IF copy% fn$="Arcopy" ELSE fn$="Ar
Move"
 2410 ans$=FNcheck("Are you sure?")
 2420 IF LEFT$(ans$,1)<>"Y" AND LEFT$(an
s$,1)<>"y" PROCrehead:ENDPROC
 2430 PROCmessage("Area "+copy$)
 2440 MOVE x2,y2:MOVE x1,y1:PLOT 188+are
acode%,x,y
 2450 PROCrehead
 2460 ENDPROC
 2470 :
 2480 DEF PROCliner
 2490 lastJ%=3
 2500 fn$=arrow$
 2510 IF repeat% PROCmessage("Repeating
line") ELSE p1=FNinput("Length?")
 2520 IF NOT repeat% p2=FNinput("Angle (
deg)?"):IF ABS(p2)>360.9 PROCrehead:ENDP
ROC
 2530 IF NOT repeat% linelen=p1/scale:li
neang=RAD(p2)
 2540 x5=x+linelen*COS(lineang):y5=y+lin
elen*SIN(lineang)
 2550 MOVE x,y:PLOT linecode%+dot%,x5,y5
 2560 IF NOT fix% THEN x=x5:y=y5:J%=13
 2570 IF repeat% PROCdelay
 2580 PROCrehead
 2590 ENDPROC
 2600 :
```

```
 2610 DEF PROCcirc
 2620 lastJ%=4
 2630 fn$=circ$
 2640 IF repeat% PROCmessage("Repeating
circle") ELSE p1=FNinput("Radius of circ
le?"):IF p1=0 PROCrehead:ENDPROC
 2650 IF NOT repeat% circrad=p1/scale
 2660 MOVE x,y:PLOT shapcode%+fill%+144,
x+circrad,y
 2670 IF repeat% PROCdelay
 2680 PROCrehead
 2690 ENDPROC
 2700 :
 2710 DEF PROCmove
 2720 IF J%<>7 ENDPROC
 2730 curs$="MOVE":PROCheading
 2740 curscode%=4:copycode%=5:linecode%=
5:shapcode%=5
 2750 ENDPROC
 2760 :
 2770 DEF PROCdraw
 2780 IF J%<>8 ENDPROC
 2790 curs$="DRAW":PROCheading
 2800 curscode%=5:copycode%=5:linecode%=
5:shapcode%=5
 2810 ENDPROC
 2820 :
 2830 DEF PROCrubout
 2840 IF J%<>9 ENDPROC
 2850 curs$="DEL ":PROCheading
 2860 curscode%=7:linecode%=7:shapcode%=
7
 2870 ENDPROC
 2880 :
 2890 DEF PROCdelete
 2900 IF J%<>10 ENDPROC
 2910 fn$="DELETE":ans$=FNcheck("Are you
 sure?")
 2920 IF LEFT$(ans$,1)<>"Y" AND LEFT$(an
s$,1)<>"y" PROCrehead:ENDPROC
 2930 MOVE x1,y1:PLOT 103,x,y
 2940 PROCrehead
 2950 ENDPROC
 2960 :
 2970 DEF PROCcopy
 2980 IF J%<>12 ENDPROC
 2990 IF vector% PLOT copycode%+dot%,x1,
y1 ELSE PLOT copycode%+dot%,x,y1:PLOT co
pycode%+dot%,x1,y1:PLOT copycode%+dot%,x
1,y:PLOT copycode%+dot%,x,y
 3000 IF copycode%=5 THEN copycode%=7 EL
SE copycode%=5
 3010 ENDPROC
 3020 :
```

```
 3030 DEF PROCfill
 3040 IF J%<>22 ENDPROC
 3050 IF fill% fill%=0:fill$="Outl" ELSE
 fill%=8:fill$="Fill"
 3060 PROCheading
 3070 ENDPROC
 3080 :
 3090 DEF PROCcolour
 3100 IF J%<>23 ENDPROC
 3110 IF col% col$="Back":col%=FALSE:GCO
L 0,129:GCOL 0,0 ELSE col$="Fore":col%=T
RUE:GCOL 0,128:GCOL 0,1
 3120 PROCheading
 3130 ENDPROC
 3140 :
 3150 DEF PROCdump
 3160 IF J%<>33 ENDPROC
 3170 fn$="Dump  ":PROCheading
 3180 VDU 2,21
 3190 @%=&00020101:PRINT '"Scale factor
is "; scale '"Printed scale is "' SPC(10
); scale*56.705;" units to one cm"
 3200 VDU 6,3
 3210 TIME=0:REPEAT:UNTIL TIME>t%
 3220 *BPRINT P5 F I W2 L100
 3230 PROCrehead
 3240 ENDPROC    -
 3250 :
 3260 DEF PROCnoproc
 3270 PROCmessage("Procedure not yet inc
luded")
 3280 PROCdelay:PROCdelay
 3290 PROCrehead
 3300 ENDPROC
 3310 :
 3320 DEF PROCnone
 3330 PROCmessage("No function on this k
ey")
 3340 PROCdelay:PROCdelay
 3350 PROCrehead
 3360 ENDPROC
 3370 :
 3380 DEF PROCdelay
 3390 LOCAL T%
 3400 T%=TIME
 3410 REPEAT UNTIL TIME>T%+150 OR ADVAL-
1
 3420 ENDPROC
 3430 :
 3440 DEF PROCerror
 3450 *FX 4
 3460 REPORT:PRINT " at line ";ERL'
 3470 ENDPROC
```

Designing Screen and Printer Characters

Eddy Hunt explains how to define alternative character sets for both screen and printer as a follow-up to last month's article on using foreign character sets with View.

In an article in last month's BEEBUG (Vol.7 No.4), I described some foreign language character sets for use with View, Acorn's word processor. Unfortunately, there always seems to be one occasion where no matter what character set you have, there is still a need to print something different. This article will describe how to define your own characters. Once they have been designed, you can install them on your machine using the program listed last month.

Although the ideas described here were developed for use with View, they could be used in any context which allows user-defined characters. Note also that the coding given here for a printer is specific to the Epson, but that the principles will be the same for other makes, and details can be found in your printer manual.

Last month's system uses a common program, with different sets of DATA statements, with each set numbered from line 9000 onwards. Each set of DATA statements holds a particular set of character definitions. The first listing this month shows the character definitions for the Turkish language, and we will use this as an example to explain how the system works. You can then follow this when defining your own character sets.

The DATA statements are enclosed in quotation marks so that if a comma appears in a character definition it will not be read as two separate statements separated by the comma. Although it is not essential to place each character definition on a separate line, it does make reading them easier. The first character in the DATA statement indicates which key must be pressed to generate the corresponding character, and the end of the statement can be used to include a comment about the character being defined (the comment consists of all the characters following the second hyphen).

```
9000DATA "+^3C028002000280023C-660066666666663C-U-umlaut"
9010DATA "Q^1CA20022002200A21C-423C66666666663C-O-umlaut"
9090DATA "xd384401440146004400-003C6660663C1838-c-cedilla"
9095DATA "wd205401540156005408-003E603C067C1838-s-cedilla"
9140DATA "q^001CA2002200A21C00-42003C6666663C00-o-umlaut"
9150DATA ";^003C80020002803C02-6600006666663E00-u-umlaut"
9200DATA "<^0000000C010E000000-0000000000181830-comma"
9300DATA "@d182481248124813E00-3E003E66663E063C-soft-g"
9310DATA ">^000022803E80220000-18007E3C3C3C3C7E-I-dot"
9540DATA "W^609400950095029448-3C66603C463C0818-S-cedilla"
9550DATA "X^784401840186008448-3C666060663C0818-C-cedilla"
9600DATA "^^020550159005500E01-1824003C063E663E-a circumflex"
9610DATA ",^0022003E0002000000-0000381818183C00-i no dot"
9620DATA "~^0052009E0042000000-1824003818183C00-i circumflex"
9630DATA "|^1EA100A104A104A116-7E003C66606E663C-soft-G"
9999DATA **
```

Turkish Character Definitions

The first group of characters, up to the first hyphen, contains the codes for the printer, and the second group is for the definition of the screen character. Thus, in principle, each definition follows the format:

```
<char><printer defs>-<screendefs>-<comment>
```

Here, 'char' is the character to be redefined, and 'printer defs' and 'screen defs' are the character definitions as described below.

DEFINING THE PRINTER CHARACTERS

The printer prints up to 11 possible horizontally separated positions per character, although only 9 are used in order to permit spaces between adjacent characters. Along each of the 9 vertical lines, points can be printed in 9 positions, although the top and bottom positions cannot both be used, thus a vertical grid of eight is sufficient. If the bottom points are used, 'descenders' are said to be printed. Details can be found in the Epson manual under the instruction ESC-&.

The character '^' in the second position of a definition indicates that descenders are *not* used. The letter 'd' (or any other character) indicates that descenders will be printed. The next 18 characters are hexadecimal digits indicating the position of the dots on each vertical line.

To form the code for a printer character, draw a rectangle 9 squares horizontally, by 8 squares vertically. Place an 'x' in each of the boxes in the position where you want the dots to occur (horizontally adjacent boxes cannot both be printed, so you should avoid putting an 'x' in both of them - see your printer manual for more information on this). The character will appear much more horizontally elongated than it will when printed. Now below each vertical line write the hexadecimal codes corresponding to the eight boxes (the top line is the most significant) reading from top to bottom. The nine pairs of hexadecimal digits form the next 18 characters of the definition. See Figure 1 for an example of the coding for the Turkish dotted letter 'I'.

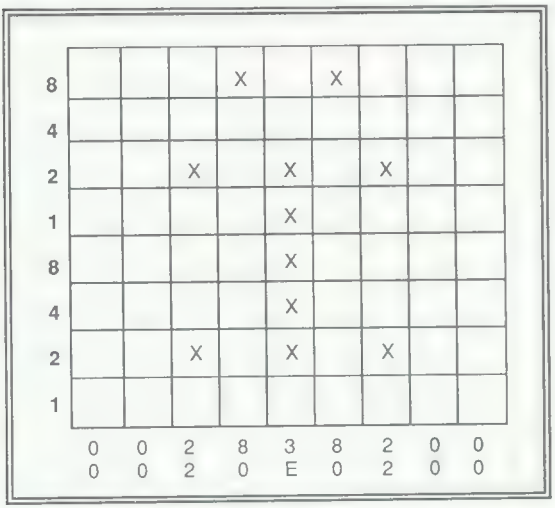


Figure 1. Designing a printer character

DEFINING THE SCREEN CHARACTERS

The hyphen is not read as such, and hence any other character could be substituted in its place. The next group of characters consists of eight pairs of hexadecimal digits describing the appearance of the character on the screen. Details can be found in the User Guide under the VDU 23 command for the format of user-defined characters. The screen character is printed using an 8x8 grid, with each pair of hexadecimal digits describing one horizontal group of eight points, starting at the top and working down to the bottom. The remaining characters in each DATA statement definition are not read, but identify each character by name.

TESTING THE RESULTS

The second listing is a utility program which eases the checking and correcting of DATA statements. These must be appended to the test program for this purpose. It is best to spool out the DATA statements using *SPOOL and save them in that form. You can then use *EXEC, either to append them to the test utility to check that they are correct, or to the base program published last month to install the definitions on your system. Using the test program, the printed and screen character definitions are shown on the screen for checking.

Three keys are used to control this program. Typing 'L' prompts for a line number - enter that of one of your DATA statement definitions. Entering 'N' moves on to the next DATA

statement, while 'Q' allows you to quit from the program.

In this way a stock of character definitions can be built up. These definitions may be used with the program as published last month, or similarly in your own programs.

```
   10 REM Program CharTest
   20 REM Version B1.0
   30 REM Author  Eddy Hunt
   40 REM BEEBUG  October 1988
   50 REM Program subject to copyright
   60 :
  100 MODE 6:ON ERROR GOTO 260
  110 Quit%=FALSE:VDU23,1,0;0;0;0;
  120 PRINT'"CHARACTER DISPLAY"
  130 PRINT'"The printed and screen cha
racters for"'"each DATA statement can be
 displayed."
  140 DIM pc$(1):pc$(0)=" ":pc$(1)="*"
  150 DIM sc$(15),pb%(15)
  160 sc$(0)="    ":sc$(1)="   *":sc$(2)
= "  * ":sc$(3)="  **"
  170 sc$(4)=" *  ":sc$(5)=" * *":sc$(6)
=" ** ":sc$(7)=" ***"
  180 sc$(8)="*   ":sc$(9)="*  *":sc$(10
)="* * ":sc$(11)="* **"
  190 sc$(12)="**  ":sc$(13)="** *":sc$(
14)="*** ":sc$(15)="****"
  200 REPEAT
  210 PROCnext:CLS:READ cd$
  220 IF cd$="***" PRINT"End of data":Qui
t%=TRUE ELSE IF NOT Quit% THEN PROCcgen:
PROCinfo
  230 UNTIL Quit%
  240 END
  250 :
  260 IF ERR=41 THEN PRINT"No such line"
:G%=INKEY(100):GOTO 200
  265 MODE 7
  270 IF ERR=42 THEN PRINT"No DATA state
ments" ELSE REPORT:PRINT" at line ";ERL
  280 END
  290 :
 1000 DEF PROCcgen
 1010 rc%=ASC(LEFT$(cd$,1))
 1020 ND%=MID$(cd$,2,1)="^":p%=3
 1030 FOR i%=0 TO 8
 1040 pb%(i%)=EVAL("&"+MID$(cd$,p%+2*i%,
2))
 1050 NEXT
 1060 PROCprchr
 1070 p%=p%+19
 1080 PROCscchr
 1090 ENDPROC
 1100 :
 1110 DEF PROCprchr
 1120 xs%=1:ys%=1
 1130 FOR x%=0 TO 9
 1140 cl%=0:pv%=128
 1150 FOR y%=0 TO 7
 1160 cn%=pb%(x%) DIV pv%
 1170 cl%=cn%
 1180 IF cn%=1 THEN pb%(x%)=pb%(x%)-pv%
 1190 pv%=pv% DIV 2
 1200 PRINTTAB(xs%+x%,ys%+y%)pc$(cn%)
 1210 NEXT
 1220 NEXT
 1230 ENDPROC
 1240 :
 1250 DEF PROCscchr
 1260 xs%=20:ys%=1
 1270 FOR i%=0 TO 7
 1280 pb%(i%)=EVAL("&"+MID$(cd$,p%+2*i%,
2))
 1290 PRINTTAB(xs%,ys%+i%)sc$(pb%(i%)DIV
16);sc$(pb%(i%)MOD16)
 1300 NEXT
 1310 ENDPROC
 1320 :
 1330 DEF PROCinfo
 1340 VDU 23,224,pb%(0),pb%(1),pb%(2),pb
%(3),pb%(4),pb%(5),pb%(6),pb%(7)
 1350 PRINTTAB(0,10)"Screen Character =
";CHR$224
 1360 PRINTTAB(0,12)"Character Name=";MI
D$(cd$,39)
 1370 ENDPROC
 1380 :
 1390 DEF PROCnext
 1400 VDU23,1,1;0;0;0;
 1410 PRINTTAB(0,19)"Line Number/Next Li
ne/Quit(L/N/Q)"
 1420 PRINT SPC30'SPC30;CHR$13;CHR$11;CH
R$11
 1430 REPEAT:G%=GET AND NOT 32:UNTIL INS
TR("LNQ"+CHR$13,CHR$G%)
 1440 Quit%=(G%=81)
 1450 IF G%=76 THEN INPUTTAB(0,20)"Line
Number="ln%:RESTORE ln%
 1460 VDU23,1,0;0;0;0;
 1470 ENDPROC
```

A CHANGE IN THE AIR FOR MICRONET

Richard J.Brunton looks at recent changes to Micronet and Prestel, including the controversial price increases.

Micronet and Prestel have remained almost unchanged for many years now with only minor improvements and updates. Recently, however, Micronet has undertaken a series of major changes which not only affect the database itself, but also the company behind Micronet, Telemap Group Limited.

The changes began when a new Managing Director, John Tomany, was appointed to the Telemap Group. Following this appointment, staff reorganisations took place with the appointment of new Product Managers for the various areas of Micronet. This was the start of the restructuring of Micronet.

One of the first improvements was the updating and expansion of the indexing system to access pages. The effect of this was a reduction in the time taken to move around the database, and the implementation of simpler routes throughout the system.

Along with the routing changes came more frequent updating, and the introduction of new computer magazines. News stories now appear daily, as do the many updates on other sections. The magazines are updated on specific days of the week, and extensive archives store the last four issues of each on-line. All the magazines have letters, hints and tips, articles, information from experts, and some even have their own chatlines, for example the BBC section.

Previously, the computer section only covered the widely used home computers, such as the Spectrum, BBC and Commodore. Now, the area has been expanded and provides comprehensive coverage for other Acorn machines, PCs and compatibles, Commodore 64 and 128, Amiga and ST, and the Amstrad CPC and PCW. Lately, areas for Z88 and QL users have been introduced.

The Micronet company Bytemail, which specialises in hardware and software by mail-order, has been expanded to provide for the increased range of computers covered, and also caters for other computers such as the Electron.

Recently, an area called Xtra! was introduced, supplying entertainment which has little or nothing to do with computers. Xtra! includes reviews of films, news of current events, radio and television guides, and also brings together Micronet's older leisure areas.

Within the Xtra! database there are a number of features, including Shades (the Multi-User Dungeon), 20th Century Hamster (a collection of quizzes and competitions with cash prizes), Starnet (a play-by-mailbox multi-user game), Bazaar (which includes classified and lonely hearts sections, and areas for buying and selling, and for expressing your views) and finally the newest introduction for Micronet members and now for Prestel members, Teletalk.

Teletalk is very similar to Shades, yet it is not a game. Teletalk is a teleconferencing system for both private and business users who wish to contact each other by day or night. There are over thirty areas, where conferences and meetings can be held, although many private users now access them for entertainment.

Telemap has also expanded away from Micronet and allows open access to Shades on the 0898 telephone system, under the title Funtel. This in turn is based on the three-month trial system called Hotel California. The Funtel area is intended to include many of the old Hotel California's services.

Funtel is accessed in the same way as Micronet, and apart from Shades it offers many other games and competitions to provide hours of entertainment. The problem is the cost, since the cost for the 0898 system is 25p per minute off peak and 38p per minute at other times.

The other major change to Micronet has arisen through the revised time charges introduced by Prestel. Prestel now costs 7p per minute from 8am to 6pm Monday to Saturday, while at all other times the cost is 1p per minute. Previously the charges were 6p per minute, 8am to 6pm Monday to Friday and 8am to 12 noon on Saturdays, while all other hours were free of time charges.

Annual subscriptions were also increased, and membership to Micronet and Prestel now costs £20 per quarter for private users and £30 per quarter for business users. This means an increase to private users of £14 per anum and to business users of £8 per anum. Telemap has said that private users are facing a larger increase than business users, because both Prestel and Micronet felt that business users were subsidising the private users.

The good news is that access to Telecom Gold via the Interlink gateway is now cheaper and costs only £1 per month, although there is a further charging structure for the amount of text sent.

According to Micronet and Prestel, the new charges are needed to finance the additional staff and equipment necessary for the fast updating and maintenance, for future improvements to the network of computers running Prestel, and to update the baud rates currently used by the system. Prestel gave considerable thought to the new charges, but decided that a small increase of both subscriptions and time charges would be better than a large increase in one of them. Telemap has suggested that in the future a choice might be offered to pay either by subscription or by time charges.

Micronet has, fortunately, negotiated free access to its pages between the hours of midnight and 8am every day, when most use is made of the system. Unfortunately, as soon as the user wishes to send a mailbox message, charges are incurred, since the mailboxing system is contained within the Prestel pages. Teletalk and Shades have also been spared the increased rates. Users of these two areas are not constrained to the midnight to 8am time slot to avoid time charges.

Before users were officially informed of the new charges, the news was leaked from within Micronet. The result of this was considerable anger towards Prestel, and many members commenced complaining long before official confirmation was released. Irate members sent many mailboxes to Prestel and various computer magazines who hold accounts with Prestel. Letters were also sent to Prestel's head office, and there was an increased use of Prestel's Telex service!

A 'swearathon', which was widely publicised beforehand, took place on the last day of the old charges, and the participants arranged to send obscene messages on the various chatlines, thus challenging Micronet to enforce the conditions of membership and terminate many of these user's accounts. Micronet tried in vain to delete all obscene messages sent, but finally decided to close down the chatlines being mis-used.

A large number of Micronet members also threatened to leave Micronet, but even though nights are much quieter and the total usage seems to have decreased (especially on Shades, Teletalk and the many chatlines), David Rosenbaum, spokesperson for Telemap, said that they had not lost many members. He also said that they expect a small number of members to leave when their membership renewal option comes round.

Prestel and Micronet still remain the major on-line services providers. Even though the costs have increased, they are still cheaper than the main rivals in the market. With the new additions, upgrades and future plans yet to be carried out, Micronet still remains a good choice for both the private and business user.

Micronet may be contacted (verbally) on 01-278 3143.
For a free demonstration of Micronet and Prestel phone 01-618 1111 on 1200/75 baud (using the ID 4444444444 and the password 4444).
To contact the Funtel service (available on 1200/75 baud) dial 0898 100890.

Graham Crossley throws some light on that mechanical rodent connected to your user port.

Over the past few years, the use of mice has revolutionised the way users communicate with computers. However, there are many BBC micro owners who have a mouse, but have little idea of how it works, or how to read its output directly. The purpose of this article is to explain how a mouse works, and to develop a set of routines that can be used to read the mouse's position and button status from within your own programs.

THE USER PORT

Before looking at mice, we need to take a quick look at the functioning of the computer. Inside a BBC micro are two rather complex chips called Versatile Interface Adaptors (VIA's), one being the system and the other the user VIA. These chips have two 8 bit ports (A and B) and, as the name implies, they interface various devices to the micro-processor (CPU). Port A of the user VIA forms the parallel printer interface, while port B is the user port. Figure 1 shows the connections on the user port socket. The ones marked 'X' are those used by a mouse. Pins PB0-PB7 are data lines which can be read by the computer to determine mouse movement and button presses. The two pins CB1 & CB2 are a little more complicated but their function will become clear as you read on.

INTERRUPTS

When your BBC computer is running a program, or even sitting idle, the CPU is repeatedly, but only momentarily, diverted in order to run small machine code routines which perform background tasks such as scanning the keyboard, incrementing timers etc. These tasks 'interrupt' the system, and may be generated by any one of certain specific 'events'. For example, each VIA includes two timers, and the operating system uses one of these to generate an interrupt every 0.01 seconds. This is used to control processes such as the keyboard auto-repeat. Another way to cause an interrupt is by changing the logic level on either the CB1 or CB2 lines on the user port. This is very useful, as you will see later.

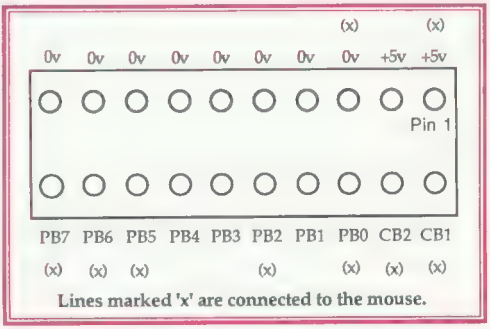


Figure 1. User port connections

To be of any practical use, we need to control what the CPU is going to do when interrupted, and in the BBC micro this control is provided by means of interrupt vectors; IRQ1V at &204/5 (high priority) and IRQ2V at &206/7 (low priority). In order to execute a specific piece of code during an interrupt we simply ensure that the address of this code is in one of the vectors prior to the interrupt occurring.

MOUSE ANATOMY

A standard mouse 'rides' on a ball which is mechanically linked to 2 rotation sensors that produce electrical pulses when the mouse, and thus the ball, are moved. This linkage is normally in the form of two rollers at right angles to each other. To stabilise a ball, there are also one or two dummy rollers which are spring loaded to hold the ball in place. The sensors are usually optical in nature, and function by rotating a slotted disc between an infra-red emitter and detector. When the roller is rotated, the disc will also turn, which will cause the infra-red beam to be interrupted

periodically. By using some simple electronics to clean up the signal from the sensor, you end up with a series of pulses as the mouse is moved. Each pulse corresponds to a fixed angular movement of the roller, and hence to a fixed linear movement of the mouse. As the two rollers are perpendicular to each other, one detects movement of the mouse in the X direction, and the other in the Y direction. Any other direction will cause both sensors to produce pulses.

So far, we have two signals. Both of these are a series of pulses, one for the X axis, and the other for the Y. But how do we tell in which direction each roller is rotating? In other words, how do you tell the difference between up and down or left and right. The solution to this is rather clever. Instead of having one infra-red sensor for each slotted disc, you have two, these being positioned at different points on the circumference (see Figure 2). Provided the relationship between the number of slots and the positioning of the sensors is correct, you can tell the direction of rotation. This is because both sensors will produce the same number of pulses, but one will start its pulse before the other. Which one is first depends on the direction of rotation. This is shown in Figure 3.

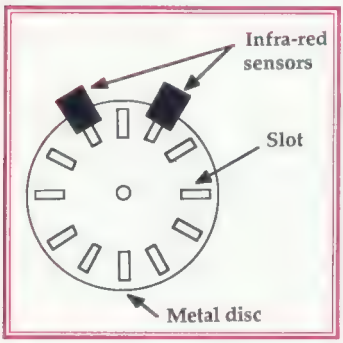


Figure 2

READING THE MOUSE

We now have four signals from the mouse which can be read by the computer to calculate the mouse's movement. The way this is done is by connecting one of the X pulse signals to the CB1 line on the user port, and one of the Y signals to CB2. As said earlier, a change in the logic level on either of these input lines can be made to cause an interrupt. There are several different ways to use these lines, but for the mouse they are programmed so that an interrupt is caused when the signal changes from logic 1 to logic 0. This is called the negative edge of the signal.

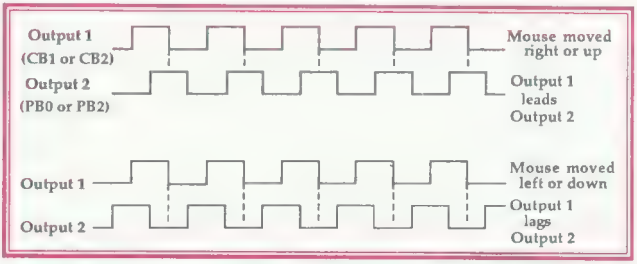


Figure 3

The other X and Y pulse signals are connected to the PB0 and PB2 lines on the user port respectively. The computer can test the logic levels on these lines simply by reading a register within the VIA. If you refer back to Figure 3, you can see that if, say, the X pulse generates an interrupt on the CB1 line, the program can immediately look at the level on the PB0 line, and use this to work out the mouse direction (logic 0 is left, logic 1 is right). The same applies to the Y movement, except that this time the interrupt is on the CB2 line, and PB2 must be checked to find the direction.

The other inputs from the mouse come from the three buttons. These are connected to the inputs PB5-PB7 on the user port (the left button is PB5, the middle one is PB6 and the right one is PB7). These inputs will normally be at a logic 1. However, when the appropriate button is pressed, the line drops to logic 0.

MOUSE DRIVER

The operating system of the BBC micro provides no handling for the mouse whatsoever. Therefore, in order to use a mouse you must add some extra software, called a mouse driver. Most mice come with such software, usually in ROM, and some packages that work with a mouse include their own mouse driver. The driver has to store the X and Y positions of the mouse, and update these as the mouse is moved. In order to do this it must be able to cope with all the interrupts produced

by the mouse. The mouse driver also has to provide some way of letting the user read the current position, and the state of the buttons.

DATA INPUT & INTERRUPT CONTROL

Before writing a mouse driver, we need to know more detailed information about how the VIA is used to read the user port. Each VIA has 16 internal registers which can be written to, and/or read from. These registers are allocated unique addresses in the computer memory map, the user VIA being at locations &FE60 to &FE6F. For the purpose of our mouse driver, only five of these actually concern us.

The first thing for a mouse driver to do is program the CB1 and CB2 lines to behave as we want. This is done using two of the VIA registers: the Auxiliary Control Register (ACR) at location &FE6B, and the Peripheral Control Register (PCR) at &FE6C. The commands to perform the re-programming from Basic are as follows:

```
?&FE6B=?&FE6B AND &FD
?&FE6C=?&FE6C AND &F
```

The reason for modifying the existing value in the registers, rather than just setting them to a certain value, is that the registers also control the other port which is used for the printer. If we accidentally changed these bits, the printer port could cease to function until the computer is reset.

Before the CB1 and CB2 lines can actually cause interrupts, they must be enabled. This is done by writing to a register at location &FE6E called the Interrupt Enable Register (IER). The value to write into the IER is &98, which will enable both CB1 and CB2 to produce interrupts. Again, any other value written could upset the other functions of the VIA.

Now, whenever the mouse is moved, an interrupt will be generated. By reading the Interrupt Flag Register at &FE6D, the source of the interrupt can be found. Bit 7 will be set to show that the VIA is interrupting, and either bit 3 or 4 will be set to show the cause of the interrupt (bit 3 for CB2, bit 4 for CB1). The other mouse signals are read from Input Register B (IRB) at location &FE60. The values of lines PB0 to PB7 are read into bits 0 to 7.

A WORKING PROGRAM

The program given here includes all the routines needed to implement a mouse driver, and uses them as the basis of a simple drawing package.

To use the mouse in your own programs, include the PROCassemble definition from this program, and call it at the start of your code. You can then activate the mouse driver using:

```
CALL setmouse
```

and when you have finished, deactivate it with:

```
CALL resetmouse
```

The X and Y co-ordinates of the current mouse position are stored in locations &70 to &73 as normal graphics co-ordinates. Your program can read these using:

```
X=!&70 AND &FFFF:Y=!&72 AND &FFFF
```

To see if a button is pressed or not, set Y% to 1, 2 or 3, for the left, middle or right buttons, and execute:

```
BUT=USRtestbuttons AND &FF
```

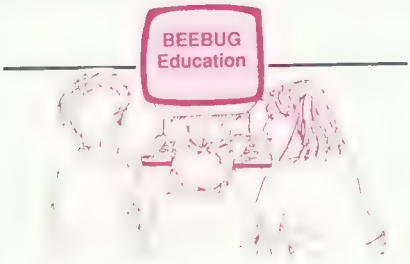
This will set BUT to zero if the button is not pressed, or to a non-zero value if it is.

The simple drawing program uses the left-hand button to draw, the middle button to change colour, and the right-hand button to clear the screen.

```
  10 REM Program Mouse Demo
  20 REM Version B1.0
  30 REM Author  Graham Crossley
  40 REM BEEBUG  October 1988
  50 REM Program subject to copyright
  60 :
 100 ON ERROR GOTO 2080
 110 MODE1
 120 PROCassemble
 130 PROCsketch
 140 END
 150 :
1000 DEF PROCassemble
1010 x%=&70:y%=&72:!x%=640:!y%=512
1020 orb=&FE60:ifr=&FE6D:ier=&FE6E
1030 acr=&FE6B:pcr=&FE6C:irqv=&204
1040 FOR opt%=0 TO 2 STEP 2:P%=&900
1050 [OPT opt%
1060 .setmouse
1070 SEI:LDA irqv:STA oldirqv
1080 LDA irqv+1:STA oldirqv+1
1090 LDA #start MOD 256:STA irqv
1100 LDA #start DIV 256:STA irqv+1
```

```
1110 LDA acr:AND #&FD:STA acr
1120 LDA pcr:AND #&F:STA pcr
1130 LDA #&98:STA ier:CLI:RTS
1140 :
1150 .resetmouse
1160 SEI:LDA oldirqv:STA irqv
1170 LDA oldirqv+1:STA irqv+1
1180 LDA #&18:STA ier:CLI:RTS
1190 .oldirqv
1200 EQUW 0
1210 :
1220 .start
1230 PHP:PHA:TXA:PHA:TYA:PHA
1240 LDA #&80:BIT ifr:BEQ exit
1250 LDA #&10:BIT ifr:BEQ up_down
1260 .left_right
1270 LDA #&01:BIT orb:BEQ mouse_left
1280 .mouse_right
1290 LDA x%:BNE incx
1300 LDA x%+1:CMP #&05:BEQ exitrgt
1310 .incx CLC:LDA x%:ADC #&04:STA x%
1320 LDA x%+1:ADC #0:STA x%+1
1330 .exitrgt JMP exit
1340 .mouse_left
1350 LDA x%:BNE decx
1360 LDA x%+1:BEQ exitlft
1370 .decx SEC:LDA x%:SBC #&04:STA x%
1380 LDA x%+1:SBC #0:STA x%+1
1390 .exitlft JMP exit
1400 .up_down
1410 LDA #&04:BIT orb:BEQ mouse_down
1420 .mouse_up
1430 LDA y%:BNE incy
1440 LDA y%+1:CMP #&04:BEQ exitup
1450 .incy CLC:LDA y%:ADC #&04:STA y%
1460 LDA y%+1:ADC #0:STA y%+1
1470 .exitup JMP exit
1480 .mouse_down
1490 LDA y%:BNE decy
1500 LDA y%+1:BEQ exit
1510 .decy SEC:LDA y%:SBC #&04:STA y%
1520 LDA y%+1:SBC #0:STA y%+1
1530 .exit PLA:TAY:PLA:TAX:PLA:PLP
1540 JMP (oldirqv)
1550 :
1560 .testbuttons
1570 LDA bitnumber,Y:BIT orb
1580 BEQ pressed
1590 .notpressed
1600 LDA #0
1610 .pressed
1620 RTS
1630 :
1640 .bitnumber
1650 EQUD &80402000
1660 ]NEXT
```

```
1670 ENDPROC
1680 :
1690 DEF FNbutton(Y%)
1700 =USR(testbuttons)AND&FF
1710 :
1720 DEF PROCsketch
1730 VDU23,128,248,240,240,248,156,14,4
;5,23,1,0;0;0;0;
1740 CALL setmouse:penx%=0:peny%=0:col%
=3
1750 VDU25,4,625;500;18,3,col%,128,25,4
,625;500;:F%=TRUE
1760 REPEAT
1770 newx%=(!x% AND &FFFF)
1780 newy%=(!y% AND &FFFF)
1790 IF newx%=1280 newx%=1279
1800 IF newy%=1024 newy%=1023
1810 IF FNbutton(1) PROCdraw ELSE PROCm
ove
1820 IF FNbutton(2) PROCcolour
1830 IF FNbutton(3) VDU16,128,25,4,penx
%;peny%;:F%=TRUE
1840 UNTIL FALSE
1850 ENDPROC
1860 :
1870 DEF PROCdraw
1880 IFnewx%=penx% AND newy%=peny% AND
NOT F% ENDPROC
1890 IF F% VDU128,25,4,penx%;peny%;18,0
,col%:F%=FALSE
1900 penx%=newx%:peny%=newy%
1910 VDU25,5,penx%;peny%;
1920 ENDPROC
1930 :
1940 DEF PROCmove
1950 IFnewx%=penx% AND newy%=peny% AND
F% ENDPROC
1960 IF NOT F% VDU18,3,col%,128,25,4,pe
nx%;peny%;:F%=TRUE:ENDPROC
1970 penx%=newx%:peny%=newy%:*FX19
1980 VDU128,25,4,penx%;peny%;128,25,4,p
enx%;peny%;
1990 ENDPROC
2000 :
2010 DEF PROCcolour
2020 col%=(col%+1)MOD4
2030 IF col%=0 THEN col%=1
2040 VDU128,25,4,penx%;peny%;18,3,col%,
128,25,4,penx%;peny%;
2050 Z%=INKEY 30
2060 ENDPROC
2070 :
2080 IF ERR=17 CALL resetmouse:MODE 7:P
RINT'"DONE" ELSE REPORT:PRINT" at line "
;ERL
2090 END
```



TOPIC WORK AND SOFTWARE
by Mark Sealey

BEEBUG Education this month looks at topic or project work for the Primary, Middle and Secondary age ranges. Since as much software for this area of the curriculum has been produced as for any other, it is only to be expected that it should be of varying quality. Before turning to some examples, it seems useful to sketch a few broad guidelines about what is available and how it can be approached.

There is very little topic software which does not fall into one of the six categories: adventure, simulation, geography, history, science, and what the publishers call 'across the curriculum'. It would be impractical to survey even 1% of what is available. You need to browse and decide whether your approach is to be an integrated one - (ignoring subject boundaries and oriented towards skills), a child centred approach, or one where you may decide to adapt something that is not quite right but falls roughly into your subject area. It is vital not to let the title or a brief product description tie you down. Good software is flexible!

To get some idea of what is available, of target age range, suitability, content and practical details, you could do a lot worse than ask to be put on the mailing list of one of the best guides, the *Rickitt Educational Software Directory*, from *Rickitt Educational Media, Ilton, Iminster, Somerset TA19 9BR.*

By and large, good software for use in topic or project work will meet three criteria. Firstly, the language on screen will not be too difficult or specialised for the users. Secondly, the material away from the computer will generate as much thoughtful activity as the programs themselves. Thirdly, the pupils will not be required to jump through a series of hoops as 'proof' of having learnt something. Software which suggests that pressing a key gets the only 'right' response should be avoided. The more open-ended an activity, the more room there is for pupils' own exploration and for them to develop skills of hypothesis formation. In general, software of the context free variety is to be preferred.

If a program has a well-defined scenario, so much the better; research suggests that learning is more effective when children are working in 'microworlds' which obey predictable and discoverable rules. Where there is a strong flavour of place, theme or time, learning is enhanced.

In history, for example, there is little to be gained in having pupils commit pages of facts to memory in order to come up with them at the right time in the program. Regrettably, the BBC micro has spawned many of these. Approaches that promote speculation about what would have happened if... or attempt to see a well-known event from an unusual point of view are usually more successful. Before using or buying project software, check that it conforms to as many as possible of the evaluation criteria offered in BEEBUG Education Vol.7 No.1.

Two suites representative of what is new in the area, and which illustrate some of the above points, are discussed here. It is to be borne in mind that very few packages satisfy every requirement, especially where applicability across the curriculum is concerned. Both *Apprentice* and *The Water Game* go quite a way towards doing so, though.

APPRENTICE
for the BBC B, B+ and Master (DFS 40T or 80T).
Supplied by Scetlander, 74 Victoria Crescent Road, Glasgow G12 9JN.
Price £21.85 inc.VAT and p&p

This suite can be used with juniors as well as secondary school pupils. Its scenario is 'small

town' life in Seventeenth Century Scotland. The pupil has to learn the crafts of archery and breadmaking, survive physical attack, site and build a mill and develop map-reading skills in a series of self-contained but connected and well-paced modules.

It is obvious from this list just how many areas of the curriculum can be brought in with imaginative teaching. The program asks pupils to arrange in order the component jobs of breadmaking. They could well use the sequencing skills needed here in other contexts such as building, farm management or games of the period.

There is another facet of topic work to be highlighted. Don't expect the computer or pupils to do all the work! Go out and buy the yeast and flour, and be prepared to get your hands wet. *Apprentice* promotes as much activity away from the computer as on it. There is even a board game concerning Burgh Government and diet and nutrition.

The package also has that reliable sign of quality: educational aims given in the excellent documentation, plus evidence of 'expert' advice and a book list for teachers and pupils. There is also a way for teachers to preview the path through the package without having to 'play each game'. When they do, they will see how the suite could very easily be adapted to a whole variety of classrooms and styles of teaching.

Project software should be usable by pupils of varying abilities; this is obviously difficult to achieve and quite rare. *Apprentice* scores by containing activities which will challenge a fair range of experiences and aptitudes related, for example, to number work and the difference between 'divergent' and 'convergent' thought. This is really quite an important distinction in the way in which 'problems' are perceived and solved by pupils. Much topic software pays no attention to the variety of ways in which children may tackle tasks.

There is probably as much as a term's work in this pack. It also provides enough background information to allow pupils who become fascinated with a particular aspect of the subject matter to explore it for themselves. *Apprentice* is thoroughly recommended.

THE WATER GAME

Supplied by CWDE Software, Regents College, Inner Circle, Regents Park, London NW1 4NS. Price £15.50 inc.VAT and p&p

This pack, aimed at slightly younger children, is every bit as impressive, although not so large, which shows that quality and size are not necessarily related. Topic software in general is effective because it is simple.

The Water Game has a simple idea - to teach pupils how to manage what is a very scarce resource in most parts of the world (lots of relevant geographical data is given in the documentation - as well as an excellent resource list).

In that sense, it could be described as specialist software. But, like *GRASS* (the database from the same producers, reviewed in BEEBUG Vol.5 No.9) it has a very wide application. After the initial menu screen (which can be recalled at any time), pupils (from 2 to 5 in number) are asked to change any of the variables in the game. These include duration and some slight changes in playing conditions.

Previously saved games can now be reloaded, and joint decisions have to be made about what the water requirements will be: for drinking, for animals, irrigation and so on. The remainder of the game uses attractive graphics and the sort of sound effects that will probably not become tiring. You have to collect the precious water and then allocate it, both in a visual game and conventional problem-solving situations.

The only small criticism of this suite, which applies to *Apprentice* as well, is an over-reliance on text. Younger children are not always able to read multiple text screens, however appealingly presented. It would,however, be churlish to make too much of this when faced with such a well thought-out compromise between real-time and 'take-your-time' activities.

PRINT FORMATTING

If your program produces large quantities of numerical output then Bernard Hill's print formatting function may be just what you need.

Many users have found that when it comes to printing or displaying numbers in any quantity, getting just the right format or alignment is not that easy. Using either the comma or semi-colon as a separator, and adjusting the value of the system variable @% should provide all that you need, but often it just doesn't seem to work. In fact, many programmers forsake these methods altogether, convert all their numbers into string format, and do all the formatting themselves.

```
#####.##     #####.##     #####.##     #####.##     #####.##     #####.##
    1.00        -3.14         9.87       -31.01        97.41      -306.02

+#####.##    +#####.##    +#####.##    +#####.##    +#####.##    +#####.##
   +1.00       -3.14        +9.87       -31.01       +97.41      -306.02

00000.##     00000.##     00000.##     00000.##     00000.##     00000.##
00001.00    -0003.14     00009.87     -0031.01     00097.41     -0306.02

+0000.##     +0000.##     +0000.##     +0000.##     +0000.##     +0000.##
+0001.00    -0003.14     +0009.87     -0031.01     +0097.41     -0306.02
```

There is a better solution, and it's not new either. In the past, many implementations of Basic provided a so-called PRINT USING function. This enabled the user to specify precisely what format any number should be printed in by effectively 'painting' a picture of how the number should be laid out. For example, a format of +##.#### would indicate that an eight character field would be used for printing. The '#' characters indicate where the digits will be placed, and the full stop where the decimal point should go. Lastly, the '+' at the front of the format simply ensures that either a plus or minus sign is printed at the beginning of the number, rather than just a space in the case of a positive value.

The program listed here contains a PRINT USING function which you can incorporate into your own BBC Basic programs. This is in the form of function called FNusing, and it is the code for this (from line 30000 to 30320) which should be added to any of your own programs which need to use this facility. Included with the function you will find a demonstration program which shows clearly all the different formats which are possible. Part of that output is reproduced here. The demo also shows clearly how to call FNusing to display or print a number.

To see the results, type the whole program in and save it before proceeding further. When you run the program a comprehensive test display will be produced. If you have a printer, pressing Ctrl-B just before you run the program will produce a more permanent record of the formats available.

The features of a print format are made up from the following code letters:

#	indicates the position for a digit.
.	indicates the position of the decimal point.
^^^^	shows where the exponent should be if 'scientific' notation is required.
,	a comma appears as a comma, for use with very large numbers.
0	same as # except that all digit positions are filled (with zeros) if necessary, rather than spaces, if the number does not fully fill the print field.
()	brackets around the format indicate that negative numbers should be enclosed in brackets (often used for financial figures) rather than printed with a minus sign.

Examining the display produced by the test program should show you exactly how to use these print formats. Now you have no excuse for producing badly formatted numeric displays. So why not add a print using function to your own programs?

```
   10 REM Program FNusing Test
   20 REM Version B1.0
   30 REM Author  Bernard Hill
   40 REM BEEBUG  October 1988
   50 REM Program subject to copyright
   60 :
  100 MODE3
  110 PROCtest("#####.##",-PI)
  120 PROCtest("+####.##",-PI)
  130 PROCtest("00000.##",-PI)
  140 PROCtest("+0000.##",-PI)
  150 PROCtest("##.###^^^^",0)
  160 PROCtest("##.###^^^^",-PI)
  170 PROCtest("##.###^^^^",-1/PI)
  180 PROCtest("###.#^^^^",-PI)
  190 PROCtest("&######",-PI)
  200 PROCtest("&000000",-PI)
  210 PROCtest("(###.##)",-PI)
  220 PROCtest("(0000)",-PI)
  230 PROCtest("+,###,###.#",-12.3)
  240 PROCtest("#,###,###.#",21.1)
  250 PROCtest("#,###,###",21)
  260 PROCtest("(#,###,###)",-10)
  270 END
  280 :
  290 DEFPROCtest(s$,X)
  300 LOCAL i
  310 PRINT
  320 FOR i=0 TO 5:PRINTTAB(13*i);s$;:NE
XT:PRINT
  330 FOR i=0 TO 5
  340 PRINTTAB(13*i);FNusing(s$,X^i);
  350 NEXT
  360 PRINT
  370 ENDPROC
  380 :
```

```
30000 DEFFNusing(f$,n)
30010 LOCAL @%,p$,x$,y$,a$,b$,n$,D,Z,E,C
,B,P,L,N
30020 p$=" ":L=LENf$:C=INSTR(f$,",")>0
30030 D=INSTR(f$,"."):IF D=0 THEN n=INT(
n+.5)
30040 Z=INSTR(f$,"0"):IF Z THEN p$="0"
30050 E=INSTR(f$,"^^^^"):IF E=0 THEN 300
80
30060 IF n=0 THEN p=0 ELSE p=INTLOGABSn
30070 =FNusing(LEFT$(f$,E-1),n/10^p)+"E"
+FNusing("+0#",p)
30080 IF ASCf$<>38 THEN 30130
30090 n$=STR$~n:L=L-1
30100 N=LENn$:IF n<0 AND L<N THEN n$=RIG
HT$(n$,L):GOTO 30100
30110 IF Z THEN x$="&" ELSE y$="&"
30120 GOTO 30320
30130 P=ASCf$=ASC"+" AND n>=0
30140 B=ASCf$=40:IF B=0 GOTO 30180
30150 L=L-2:IF D>0 THEN D=D-1
30160 IF n>=0 THEN x$=" ":b$=" ":GOTO 30
180
30170 n=-n:b$=")":IF Z x$="(" ELSE y$="(
"
30180 @%=(&1020000+256*(L-D))*SGND+L
30190 IF Z AND P THEN ="+"+FNusing(MID$(
f$,2),n)
30200 IF Z AND n<0 THEN ="-"+FNusing(MID
$(f$,2),-n)
30210 n$=STR$n:IF NOT C THEN 30280
30220 IF ASCn$=45 a$="-":n$=MID$(n$,2)
30230 D=INSTR(n$,".")-1:IF D=-1 D=LENn$
30240 IF D<4 THEN n$=a$+n$:GOTO 30280
30250 FOR N=1 TO LENn$:a$=a$+MID$(n$,N,1
)
30260 IF (D-N) MOD 3=0 AND N<D a$=a$+","
30270 NEXT:n$=a$
30280 N=LENn$
30290 IF L<N THEN =f$
30300 IF L=N THEN =x$+y$+n$+b$
30310 IF P THEN L=L-1:n$="+"+n$
30320 =x$+STRING$(L-N,p$)+y$+n$+b$
```

B

Points Arising....Points Arising....Points Arising....Points Arising....

MULTI-COLUMN PRINTING (Vol.7 No.1)

Unfortunately the two print styles, pica and elite, were confused. Changing lines 2100 and 2110 as shown will ensure that the correct style is selected.

```
  2100 UNTIL P%=69 OR P%=80 OR P%=67:
PRINTCHR$P%':pmode%=-(P%=69)-4*(P%=67)
  2110 mlin%=80-16*(P%=69)-52*(P%=67)
```

MATHEMATICAL WORMS (Vol.7 No.4)

The number following GOTO in line 3070 should be changed to read as follows:

```
  3070 IF ABS(X%)>EX% OR ABS(Y%)>EY%
THEN C%=31:GOTO 3160
```

and not as published in the magazine. Our apologies for this small error.

B

1st COURSE

A miscellany of items compiled by Mike Williams

This month's First Course contains a number of different items. There are two more fancy scrolling routines by Lindsay Cullen, E.T.Ems explores character sorting within strings, while Lindsay Cullen (again) provides a useful visual insight into string sorting.

SCROLLING STRINGS

Lindsay Cullen

To add to last month's collection of scrolling routines, here are two more ideas for improving screen displays. The first of these is very short, and serves a different purpose to most of those we looked at previously. It takes any string (up to Basic's limit of 256 characters), and writes this on the screen at a readable speed, one character at a time. The rate at which characters are displayed can be easily controlled by the value in line 150. The larger the value the slower the rate of display.

```
100 DEF PROCdisplay(x%,y%,a$)
110 PRINTTAB(x%,y%);
120 length=LEN(a$)
130 FOR loop%=1 TO length
140 PRINTMID$(s$,loop%,1);
150 TIME=0:REPEAT UNTIL TIME>16
160 NEXT
170 ENDPROC
```

The parameters required by the procedure are the co-ordinates of the position on the screen at which text should commence, and the string itself.

If you want to display more that 256 characters then just split your text up into several strings and call the procedure for each one successively. Depending on where a string meets the left-hand edge of the screen, the occasional space between words may need to be omitted. Trial and error is the best way to sort this out.

Lindsay Cullen's other scrolling procedure is really misnamed-named as such. What it does is to display a character string in mode 7 double-height characters on the screen, and then split the two halves of the line of text (upper and lower) into two separate single-height lines, one above and one below the original position. The effect is actually much more impressive than it might sound from the description.

```
200 DEF PROCdisplay(a$,y%)
210 len%=LEN(a$)
220 x%=(36-len%)/2
230 PRINTTAB(x%-2,y%);CHR$(134);CHR$(1
41);a$
240 PRINTTAB(x%-2,y%+1);CHR$(134);CHR$
(141);a$
250 FOR loop%=0 TO len%-1
260 ch$=MID$(a$,loop%+1,1)
270 IF ch$=" " THEN 340
280 FOR move%=1 TO 4
290 PRINTTAB(x%+loop%,y%-move%);ch$
300 PRINTTAB(x%+loop%,y%+1+move%);ch$
310 PRINTTAB(x%+loop%,y%+1-move%);" "
320 PRINTTAB(x%+loop%,y%+move%);" "
330 TIME=0:REPEAT UNTIL TIME>6
340 NEXT:NEXT
350 ENDPROC
```

The text string specified is centred horizontally on the screen in a vertical position determined by the parameter y%. The separation into two separate lines of text is performed by the loop at lines 280 to 340, with the new lines being positioned 4 lines above and below the original. Changing the value '4' in line 280 will control the degree of separation, while adjusting the time value in line 330 will determine the speed at which the separation occurs.

These procedures, together with those given last month, provide a useful collection for a variety of effects when it comes to displaying text. No doubt by experimenting you could devise many more.

CHARACTER SORTING

E.T.Ems

The First Course article in BEEBUG Vol.7 No.1 gave a function for ordering the characters within a string as an example of the use of Basic's string handling functions. That original function is reproduced below.

```
 1000 DEF FNsort(string$)
 1010 LOCAL n:n=LEN(string$)
 1020 FOR i=n TO 1 STEP -1
 1030 FOR j=1 TO i-1
 1040 IF MID$(string$,j,1)>MID$(string$,
j+1) THEN string$=LEFT$(string$,j-1)
      +MID$(string$,j+1,1)
      +MID$(string$,j,1)
      +RIGHT$(string$,n-j-1)
 1050 NEXT:NEXT
```

Listed below is an alternative function, which makes a vast improvement to the speed at which this process is carried out.

```
1000 DEF FNsort(string$)
1005 LOCAL ascii,S,S$
1010 S$="":S=0
1020 FOR ascii=32 TO 126
1030 REPEAT
1040 S=INSTR(string$,CHR$(ascii),S+1)
1050 IF S>0 THEN S$=S$+CHR$(ascii)
1060 UNTIL S=0
1070 NEXT
1080 =S$
```

The routine uses the fact that when we are sorting single characters, we know in advance the range and order of all the possible elements, and hence we can build this knowledge into the program. Line 1020 is a FOR loop which takes each ASCII code from 32 to 126 in turn. The routine then uses the INSTR function to search for every occurrence of the corresponding character in the original string and assign these to a new string, which is built up and returned by the function. The other difference is that the characters are not progressively re-ordered within the string. The ordered string returned could replace the original if required. On long strings, Mr.Ems' routine takes under 3 seconds compared with over two minutes for the original routine.

In fact, the technique described above can be readily modified to sort a character string into any predefined order. What is required is a *mask* which is a string containing every character in the desired order. This then determines the basis for ordering any other string. The function below uses such a mask, to order the characters. For flexibility, the mask should be specified in the main program, and supplied to the function as a parameter.

```
1000 DEF FNsort(string$,mask$)
1010 LOCAL char$,count,S,S$
1020 S$="":S=0
1030 FOR count=1 TO LEN(mask$)
1040 char$=MID$(mask$,count,1)
1050 REPEAT
1060 S=INSTR(string$,char$,S+1)
1070 IF S>0:S$=S$+char$
1080 UNTIL S=0
1090 NEXT
```

A possible mask might be:

```
100 mask$="AaBbCcDdEeFfGgHhIiJjKkLlMmN
nOoPpQqRrSsTtUuVvWwXxYyZz0123456789 !#$%
&'<>*+,-./=}@{|~£:;()_;?\^[]"
```

This orders all the characters alphabetically irrespective of case, followed by all the digits in order, and then all the other printable characters. The three character-sorting functions are included on the magazine disc as complete working demos together with timings. You can also experiment with different masks of your own choosing.

VISUAL SORTING

Lindsay Cullen

Our final program this month is longer than we normally feature in First Course articles, but it should prove quite instructive to beginners. It provides a visual demonstration of three sorting methods:

Bubble Sort - compares, and swaps if necessary, adjacent elements so that on each pass the smallest element bubbles up to the top of the list (or as here the largest sinks to the bottom).

Selection Sort - takes each position in turn and searches for the smallest element in the remainder of the list to fill this position.

Cullen's Sort - a variation on the Selection Sort by the author of the program which swaps the element in each position with the first smaller element found.

In each case, you are asked to enter up to 20 names, which are then sorted. The names are displayed on the screen and as you watch you can see just how the names are swapped around. The program asks for a 'pause' value: 0 gives the fastest display, larger values (1, 2, 3 etc.) slow the display down. At the end, pressing the space bar returns to the menu display.

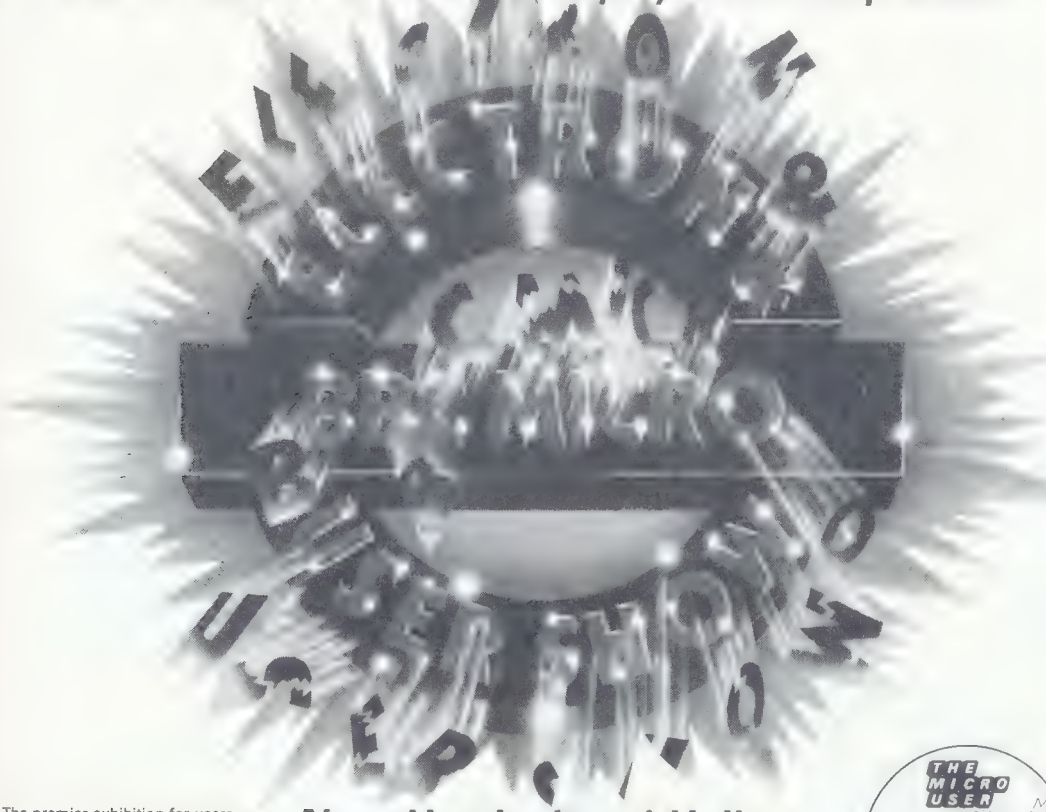
We expect to cover sorting (and searching) techniques in more detail in a future Workshop series.

```
  10 REM Program Sorts
  20 REM Version B1.4
  30 REM Author  Lindsay Cullen
  40 REM BEEBUG  October 1988
  50 REM Program subject to copyright
  60 :
 100 MODE7:ON ERROR GOTO 280
 110 VDU23,1,0;0;0;0;
 120 DIM name$(20),he$(3)
 130 he$(1)="Bubble Sort"
 140 he$(2)="Selection Sort"
 150 he$(3)="Cullen's Sort"
 160 REPEAT:PROCmenu
 170 CLS:PROCdouble(4,1,he$(key%))
 180 PROCnames
 190 PROCshiftnames
 200 IF key%=1 PROCbubble
 210 IF key%=2 PROCselect
 220 IF key%=3 PROCcullen
 230 PRINTTAB(0,20);"Finshed"
 240 key%=GET
 250 UNTIL FALSE
 260 END
 270 :
 280 MODE7:REPORT:PRINT" at line ";ERL
 290 END
 300 :
1000 DEF PROCdouble(x,y,m$)
1010 PRINTTAB(x-1,y)CHR$141;m$
1020 PRINTTAB(x-1)CHR$141;m$
1030 ENDPROC
1040 :
1050 DEF PROCmenu
1060 hed$="Sort Demonstrations"
1070 VDU26:CLS:PROCdouble(10,0,hed$)
1080 FOR i=1 TO 3
1090 PRINT''TAB(5);STR$(i);".";he$(i)
1100 NEXT i
1110 REPEAT:key%=GET-48
1120 UNTIL key%>0 AND key%<4
1130 ENDPROC
1140 :
1150 DEF PROCnames
1160 PRINT'"Please type in up to 20 nam
es, each no"'"longer than 7 letters long
."'"Press Return to exit."'
1170 VDU28,0,24,39,8
1180 VDU23,1,1;0;0;0;:loop%=0
1190 REPEAT:loop%=loop%+1
1200 PRINT"Name"+STR$(loop%)+":"
1210 REPEAT
1220 PRINTTAB(8,VPOS-1)STRING$(15," ")
1230 INPUTTAB(8,VPOS-1) a$
1240 UNTIL LEN(a$)<8
1250 IF a$<>"" THEN name$(loop%)=a$
1260 UNTIL a$="":max%=loop%-1
1270 INPUT'"Pause Please: ",PA
1280 VDU23,1,0;0;0;0;
1290 ENDPROC
1300 :
1310 DEF PROCbubble
1320 REPEAT:finish%=TRUE
1330 FOR b%=1 TO max%-1
1340 IF name$(b%)>name$(b%+1) THEN PROC
swap(b%,b%+1):finish%=FALSE
1350 NEXT
1360 UNTIL finish%
1370 ENDPROC
1380 :
1390 DEF PROCshiftnames
1400 VDU28,0,24,39,3:CLS
1410 FOR x%=0 TO 26:FOR i%=0 TO max%
1420 PRINTTAB(0,i%);name$(i%);SPC25;TAB
(x%,i%);name$(i%)
1430 NEXT:NEXT
1440 ENDPROC
1450 :
1460 DEF PROCswap(a%,c%)
1470 PROCleft(a%,17)
1480 PROCleft(c%,9)
1490 PROCuandd(a%,c%,1,9,-1)
1500 PROCuandd(c%,a%,-1,17,1)
1510 PROCright(a%,17,8,c%)
1520 PROCright(c%,9,16,a%)
1530 a$=name$(a%)
1540 name$(a%)=name$(c%):name$(c%)=a$
1550 ENDPROC
1560 :
1570 DEF PROCleft(pos%,to%)
1580 name$=LEFT$(name$(pos%)+STRING$(7,
" "),7)
1590 FOR x%=1 TO to%
```

Continued on page 64

It's back!

The show that ALWAYS keeps you one step ahead



The premier exhibition for users of all Acorn machines returns to its popular venue in the heart of the capital.

Traditionally the liveliest event of the year on the Acorn calendar, the pre-Christmas show is the one you just cannot afford to miss.

It's your value-for-money passport to:

- 70 exhibitors displaying all the latest developments across the entire Acorn range.
- Archimedes World – which provides a fascinating glimpse into the current and future roles for this remarkable machine.
- Technical advice from the UK's leading experts on all Acorn computers.
- Hundreds of special offers for the BBC Micro and Electron waiting to be snapped up as top-value Christmas presents.

All this – and so much more – at the 20th record-breaking Electron & BBC Micro User Show.

You can even save yourself £1 before you get there by using this advanced ticket form.

New Horticultural Hall, Greycoat Street, London SW1

10am-6pm Friday, November 11
10am-6pm Saturday, November 12
10am-4pm Sunday, November 13

ELECTRON & BBC MICRO USER SHOW

Advance ticket order

SAVE £1

Please supply tickets for November show:

☐ Adult tickets at £4 (save £1) ..... £

☐ Under-16s tickets at £2.50 (save £1) ..... £

☐ Cheque enclosed made payable to Database Publications Ltd. Total £ ____

☐ Please debit my credit card account: ☐ Access ☐ Visa Expiry date __/__

Name

Address

Signed

Post to: *Database Exhibitions, Europa House, Adlington Park, Adlington, Macclesfield SK10 4NP*

Admission at door
£5 (adults)
£3.50 (under 16s)
Advance ticket orders must be received by November 2, 1988

DATABASE EXHIBITIONS

PHONE ORDERS Ring Show Hotline 0625 879920
Prestel Orders KEY *89 THEN 614568383
MicroLink/Telecom Gold Orders. 72 MAG001
Please quote credit card number and full address

A493

THE MICRO USER INNOVATION ROW 1988

Take a stroll down Innovation Row – a brand new show feature area, specially constructed for the event.

See the grand finalists displaying their breakthroughs in public for the first time. And you can help pick the winners by casting a vote in both categories of the awards – BBC Micro and Archimedes.

How to get there

Underground: The nearest tu stations are VICTORIA (Victori District and Circle Lines), ST. JAMES'S PARK (District a Circle Lines) and PIMLICO (Victoria Line).

By British Rail: VICTORIA STATION. The halls are a 10-minute walk from the station

By Bus: 11, 24, 29, 70, 76 and Red Arrow 507 to Victoria Stre – alight Army and Navy Store

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

WANTED Secondhand 40/80T Midwich Disc Drive at resonable price. Tel. 0670716109.

Lots of games on cassette and disc. Complete Micro User and Acorn User from Jan 85 to July 88 with binders, £13 per volume. Phone for list of games prices. Uxbridge (0895) 31163.

WANTED InterBase ROM and documentation for BBC B. Tel. (0332) 556381.

For BBC B: Fleet Street Editor £20, Digimouse and software £15, both for £30. Voyager 7 modem £40, Commsoft Comms Software £15, both for £50. For Electron. Quickshot II joystick controller plus software £5. Life of Repton £2.50, Starship Command £0.75, Desk Diary £0.75, Me and My Micro £0.75, Tel. Tonbridge (0732) 359959.

1770 Disc Interface £35. Allophone speech synthesiser (not Acorn) £15. Both unused, in mint condition. Tel. (0294) 52250 after 6pm.

BBC Model B with ATPL ROM board/Sideways RAM, Acorn 6502 2nd Processor , dual disc drives, Opus hires colour monitor, many ROMs and lots of software, £600. Tel. (0886) 32083.

Acorn Electron with Plus 1 and ACP Plus 4 interface, View wordprocessor and ViewSheet cartridges. £120 ono. Brother M1009 (GLP) printer £60 ono. All come with original documentation and all are in excellent working order. Tel. 021-378 1721 after 6pm.

Canon Typestar 5r (portable typewriter /serial printer) plus cable to BBC, 4 new ribbons, hardly used £110 ono. Four black spare pens for Silver Reed EB 50/JB 10 plotter £1. Wordwise Plus (1.4E), Watford WorkAid, BEEBUG Dumpmaster II - above ROMs £12 each. Gemini Office, Office master, ViewChart disc, ViewIndex disc - £5 each. Unused parallel printer cable £3. Tel. (0532) 651614.

Master 128, Cumana 40/80T DS/DD PSU disc drive, Exmon II, Interword, AMX Superart, Dumpout 3, Icon Master, manuals 1&2, data recorder, joystick, covers, leads, discs, software BEEBUG Vols.2-6, A/U mags, extra 32k SWR cartridge (write protected) other books, bargain £450. Tel. (0375) 380369.

Twin Cumana 40T SS disc drive £100, Master single plinth £5, Microvitec medium resolution colour monitor £125. Tel. (056 888) 301 after 7pm.

Solidisk 256k 4Meg board plus 32k Manager ROM plus Software - £80, Interword ROM £35, Both £100. Tell. (041-641) 1200.

BBC accessories - Torch Graduate 256k plus twin disc drives plus MS-DOS and Xchange software £250. Colour monitor RGB/video/sound £120. Twin 40/80T drives with integral PSU £115. Tel. (0452) 617725 evenings.

Acorn Z80 second processor (incl Business Software) £150. Cumana touch pad (including software) £25. Epson 40 30mm roll printer (incl spare roll) £25. Computer Village CVx 16-2 ROM/RAM expansion board £25. Original tape and disc software from 50p upwards. Please send for list. Tel. (0325) 321719.

Teac 40T DS disc drive £35, Prism 1000 modem with Micronet software on ROM £25, Watford 13 ROM board £10, Wordwise Plus £20, Watford ROM Manager £10, Spellcheck II £10, BEEBUG toolkit £10, Vine TD ROM £5, all with manuals etc. Will swop for 80T DS disc drive. Tel. 091-237 0536 after 6pm.

BBC micro issue 7 board without DFS £175. RML 480Z Link Machine complete with dual disc drive suitable for network. Software is considerable - includes PASCAL, £350. Amiga with high resolution screen and second disc drive- no software £350. Tel. 01-989 2640.

WANTED: Have you the video quantisation kit to supply, for a fee, an Arc screen dump of a photo? Contact Des Fisher Te. (0236) 20199 (day) or (o786) 833541 (evening).

Four-pen colour printer/plotter (in excellent condition) Tandy model CGP!!% with manual, pens and paper £50. Tel. (03302) 2949 evenings.

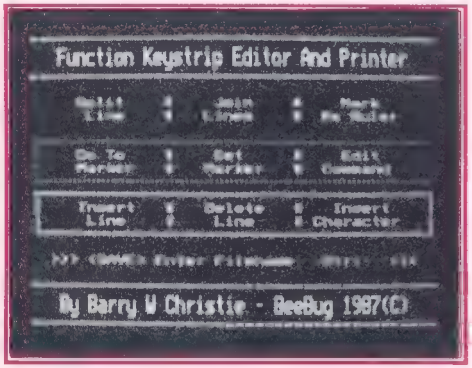
BBC Master 128 £300, Midwich single 400k 40/80T disc drive £90, Microvitec 1451 DS colour monitor £90, Taxan KP 810 dot matrix printer £150, Fischertechnik 30554 computing kit for Acorn/Model B £100 ono, ViewStore £25, Murom £11, Romit £13, Beebfont £12, ViewSheet £25. All above with manuals etc. Tel. (0837) 840718.

Continued on page 38

The Best of BEEBUG

The very best of all the programs published in BEEBUG are now available conveniently grouped together by subject on disc. The first two compilations are GENERAL UTILITIES and APPLICATIONS. In each case the disc is complete and self contained. All the programs may be loaded ready for use through a customised menu system, and all the information you need to know is included on the disc too for displaying on your screen or outputting to your printer. These discs offer exceptional value for money, and are available only to BEEBUG and RISC User members.

Each disc (5" only) costs just £5.75, plus post and packing 60p (90p for two). Simply complete the order form below, or if you wish to pay by Access, Visa or Connect just phone in your order.



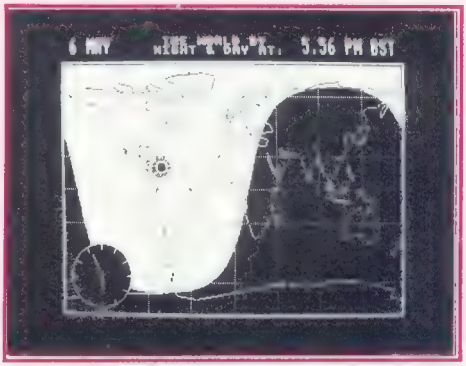
General Utilities

Printer Buffer *
ROM Controller
Sprite Editor/Animator
Multi-Character Printer Driver for View
Mode 7 Screen Editor
Multi-Column Printing
Epson Character Definer
ROM Filing System Generator
BEEBUG MiniWimp †

* Master series only.
† Requires sideways RAM.

Applications

Business Graphics
Video Cataloguer
World by Night and Day
Phone Book
Page Designer
Personalised Letter-Heads
Mapping the British Isles
Selective Breeding
Appointments Diary
The Earth from Space
Personalised Address Book



Please rush me my Best of BEEBUG discs:

1.General Utilities Disc Code 1605a ❑

2.Applications Disc Code 1604A ❑

Name _____

Address _____

_____

Memership No _____

Total £_____

Postage £_____

Grand Total £_____

I enclose a cheque for £_____. OR please debit my Access, Visa or Connect account, Card No_____/_____/_____/_____ Expiry____/____ Signed_____

Return to BEEBUG Ltd, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX. Telephone (0727) 40303.

UK BULLETIN BOARDS

All the above listed boards were believed to be active as of late July 1987. Circumstances may have changed since then and BEEBUG would appreciate members letting us know of any changes they have discovered, such as boards that have ceased operating or changed times and baud rates. Also, if you know of a board that is not listed here, please let us know for the benefit of other members.

Some new boards, particularly NBBS and OBBS systems, enable Beeb owners with Commstar or the Demon Zromm to have colour. With Commstar use mode 7 and switch off the filter mask at the command screen before entering 'Chat' mode. Zromm users should use Mode 7 and *Chat instead of *Terminal. Then when logged on answer 'Yes' to the question 'Are you using software on a BBC that allows colour'.

Aberdeen ITEC 24 Hours	0224 641585 (Aberdeen) 1200/75 Viewdata
Acorn 24 Hours	0223 243642 (Cambridge) 1200/75 Viewdata
Asylum 24 Hours	01 853 3965 (London) 300/300
BABBS I 24 Hours	0394 276306 (Felixstowe) 300/300
BABBS II 24 Hours	0268 778956 (Basildon) 300/300
Basildon ITEC 24 Hours	0268 22177 (Basildon) 1200/75 Viewdata
Beeb-Tec 24 Hours	0472 276476 1200/75 Viewdata
Betelgeuse 5 24 Hours	0463 231339 300/300 & 1200/75
Bolton BBS 20.00-08.00	0204 43082 (Bolton) 300/300
Brixton ITEC 24 Hours	01 735 6153 1200/75 Viewdata
Bob's Bizarre 24 Hours	0394 279644 300/300 & 1200/1200
C-View Rochford 24 Hours	0702 546373 (Essex) 1200/75 Viewdata
Cardiff ITEC 24 Hours	0222 464725 (Cardiff) 1200/75
CBABBS 24 Hours (Ex.Thurs.)	021 430 3761 (Birmingham) 300/300
CBBS London NW 24 Hours	0895 420164 (London) 300/300
CBBS Surrey 24 Hours	04862 25174 (Surrey) 300/300
CBBS South West 24 Hours	0392 53116 (Exeter) 1200/75 & 300/300
Club 1512 24 Hours	01 204 8755 1200/75 300/300
CFC#9 24 Hours	0395 272611 (Exmouth) 300/300
CNOL 24 Hours	0524 60399 (Lancaster) 300/300

To be continued

ASTAAD

This keystrip is designed to be used with the original version of ASTAAD for the model B. Either cut out the keystrip or photo-copy this page and place under the plastic strip on your micro when using this program.

Key f0	Key f1	Key f2	Key f3	Key f4	Key f5	Key f6	Key f7	Key f8	Key f9
SAVE SCREEN ASTAAD	LOAD SCREEN ARROW	PRINTER DUMP SHAPE	SOFT ASTAAD REPEAT F2/F9	INFILL MOVE	USER DRAW	COLOUR REVERSE LINE	DELETE LINE	DELETE AREA	CIRCLE

ASTAAD

This keystrip is designed to be used with new version of ASTAAD on pages 12 to 16 of this issue. Either cut out the keystrip or photo-copy this page and place under the plastic strip on your micro when using this program.

Key f0	Key f1	Key f2	Key f3	Key f4	Key f5	Key f6	Key f7	Key f8	Key f9
PRINT SCREEN	VECTOR RECT	SOLID OR DOTTED	HOME CURSOR	MIRROR IMAGE	MARGINS	SCALE	LINK STEAMS	SAVE SCREEN	LOAD SCREEN
SOFT ASTAAD	COPY OR MOVE	LINE OR ARROW	CIRCLE OR ARC	POLYGON ELLIPSE	INFILL OUTLINE	COLOUR REVERSE	LINE FIX OR TRANS	2 OR 16 STEPS	ORIGIN REL/ABS
ASTAAD	AREA MOVE	LINE	CIRCLE	POLYGON	REPEAT F2, F3, F4	MOVE	DRAW	RUBOUT	DELETE AREA

Personal Ads (continued from page 34)

BBC (OS 1.2) Basic II, DFS, Wordwise, Pluspaint, 40T SS disc drive, tape recorder, 3 joysticks, Quset mouse, books, full set of Input (bound), BEEBUG, Acorn User, Micro User, Disc User mags plus lots of software. £695. Tel. (0530) 35439 after 6pm.

BBC B (issue 7), Watford DFS dual 40T DS disc drive, Watford 32k RAM card, Wordwise Plus, plus many other ROMs. Software inc. Mini Office 2, Masterfile 2, Printwise, discs and games, manual and books. £350 ono. Tel. (0378) 73524.

BBC B (issue 7) OS 1.2 as new, unmodified, for just £195. Tel. 01- 794 5906.

Computer Concepts Mega-3 ROM (InterWord, InterSheet, InterChart) £55, Morley Electronics ROM board AA (Master only) £35, Care Master ROM cartridge plus Master ZIF ROM cartridge £15, Peartree MR 6000 switchable ROM/RAM cartridge (4 ROM/RAM) £8. Tel. 01-494 1365.

Master 128, Microvitec 1431, Mitsi 5.25 40/40T DD, Epson 3.5 40/80T DD, DD PSU, Demon plus Eprom, software plus magazines. £500, offered complete or will separate. Tel. 021-749 2320 weekends only.

Interword £20, Toolkit II £12, Sleuth £10, Disc Doctor £8, Watford Shadow RAM £20. All boxed with instructions. Tel. (0442) 62271 evenings.

BBC B (issue 7) with 1770 DFS, Watford 13 ROM board, 16k battery-backed sideways RAM, ZIF socket and Watford 32k shadow RAM card. Includes original manual plus welcome package and cover. All in very good condition. £375 ono 01-958 7475.

Acorn Electron 64k/Turbo (Master RAM board fitted) c/w Plus One, (fitted with EXP 2.0 ROM), ROM Box, ACP Plus 4 disc interface, (DFS & ADFS) all in excellent condition, £225 ono. Tel. (0533) 363639.

BBC B Solidisk 1770 DFS/ADFS, Watford 64k RAM/ROM board c/w 16k battery backed RAM , Watford 32k Shadow RAM Board, BCPL ROM plus SAG, GXR, Wordwise Plus, Basic editor, MAX ROM, Disc Doctor, CC's Graphics ROM, Revs, 4Tracks, Frak, Knightlore, Alien & Jetpac, Return to Eden. £320. Tel. (0454) 322138.

Quest mouse, Quest Paint plus Conquest ROMS £40, Watford ROM/RAM board (128k s/ways) £35, Dumpout 3 ROM and manual £10, Intersheet ROMs and manual £15. Tel. (0384) 455066.

Juki 6000 printer with manual and four daisywheels - Elite, Courier, Helen and Mini-tile. No printer lead. Excellent conditions. £100. Tel. 031-663 8808.

Modem, Prism 2000, Pace ROM, £25, Watford ROM (for Apollo), £15, Solidisk DFDC board with 2.2J DFS ROM, £25. Leads, manuals, Tel. (058 283) 3937.

BBC disc drive 40T SS £40, BBC PSU £40, Speech Upgrade £20, View 3 £40, ViewSheet £25, GXR 'B' £20, Wordwise £15. Offers invited. Tel. 01-903 5881.

BBC Master and and 40/80T DS drive both under guarantee and immaculate. Also green screen monitor, ref. manuals 1 & 2, Master ROM, Dabhand View guide, Viglen cartridges, BEEBUG magazines and discs and much original software. £520 cash for the lot. Tel. (0602) 392554.

BBC B, DFS, Watford ROM/RAM board with 64k sideways RAM, 16k battery-backed CMOS RAM read/write protected, PMS 6502 second processor, ZIS. Also GXR, Toolkit Plus, Wordwise Plus, WordAid ROMS. Other software on disc and tape. £350 as one, will negotiate for sale of individual items. Tel. 01-388 0392.

InterChart £15, Wordpower with M4 Power Font £20, PMS Multi-Font NTQ with 4 fonts £20, Transferom £10, all as new originals with manuals. Tel. (0235) 30471 evenings.

Hardware, software and firmware for sale. EG AMX MAX £12, teletext unit £75, replay £18, Microvitec 1431 monitor £125. Send SAE for full list write to Paul Duesbury, Riverside, Woodhead Rd, Honley, Huddersfield, HD7 2PP.

WANTED - User manuals Vol. 1 & 2 for BBC Master 128. Tel. (0652) 55130.

32016 Cambridge co-processor 1Mb 10MHz FPU latest Panos with £2000 of software including GCAL, BCPL, REDUCE, GKS-UK, MATRIX3, VUWRITER and SPICE. Only £700. Tel. (0234) 750770.

Interword complete as new with box, manual and keystrip. £35. Advanced User Guide also as new £10. Tel. (0293) 541599.

Watford Electronics 32k Shadow RAM printer buffer expansion board for Model B complete with manual and ROM, as new £45. Tel. (0784) 242817 evenings.

WANTED. Interbase ROM for BBC B complete with all documentation. Tel. (0332) 556831.

Printer Shinwa CPA80 plus £75 ono. Tel. (0203) 504254.

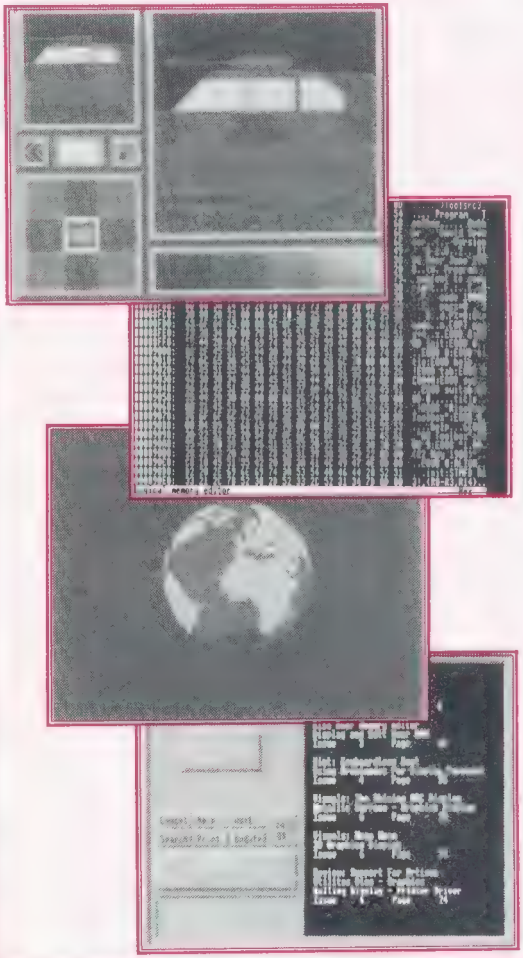
Modem, Linnet inc. command ROM, lead and unused subscription and registration. Was £139. All for £100. Tel. (0276) 65876.

RISC USER

The Archimedes Magazine and Support Group

The special disc to celebrate RISC User's 1st birthday will be available from November to all RISC User subscribers.

SPECIAL DISC CONTAINS ALL THIS FOR JUST 4.95



1. **ARCSCAN**
   A fast on-screen bibliography with powerful search facilities for all the RISC User and BEEBUG magazines. Normal price £12.

2. **PIXEL EDITOR**
   This powerful drawing tool is a full screen full-feature pixel editor for creating and editing screens and sprites.

3. **TOOLBOX**
   This incredibly useful utility features a memory editor, memory search and replace, disc editor and disassembler. TOOLBOX contains many of the features found in packages costing over £35.

4. **WORLD IN MOTION**
   A stunning animation with an oddly reminiscent feel to it.

5. **DISC MENU MODULE**
   Use the mouse to control your disc files with this extremely useful relocatable module.

6. **PRINTER BUFFER**
   This printer buffer frees your computer during long printouts and is configurable from a few bytes to 4 Mbytes. Similar to packages currently selling at £19.

Altogether the items on the disc would be worth over £50 if bought separately. See page 70 for reduced subscription rates for BEEBUG members.

THE PUBLISHER

See you at the Micro User Show in November - Stand 98.

A NO-NONSENSE, NO-GIMMICK DESKTOP PUBLISHING SYSTEM

* INTEGRATES WITH YOUR EXISTING WORDPROCESSOR
* SIMPLE BUT POWERFUL PAGE DESCRIPTION COMMANDS
* ON-SCREEN PREVIEW FROM WORDWISE, VIEW & INTER-WORD

THE PUBLISHER is a single, massive 64K ROM which holds the controlling software and 16 FONTS. Being ROM based, **THE PUBLISHER** is instantly available, no disc access required. **PREPARE, PREVIEW** and **PRINT** all from **WITHIN** your word processor **£39 + £1 P&P +VAT = £46 INC.**

CONTACT PMS FOR FULL DETAILS, SCREEN-SHOT AND SAMPLE PRINTOUT.

PMS, 38 Mount Cameron Drive, East Kilbride G74 2ES
Tel. (03552) 32796.

CHEER UP!
we've got you
covered,
with
£6.50

Patent pending and Registered Design

SEAL 'n TYPE ™

* **Protective keyboard cover through which you can type.**
* 24hr dust/spill cover
* Removable, washable, re-usable.
* **Can be custom-made for any keyboard. Ring for details.**

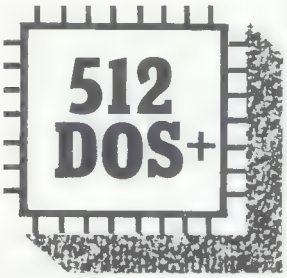
Ring or Write for our FREE catalogue

Re-inking Service £1.90	Prices are fully incl.
Ring for transporter SAE	
DMP re-inking kit £10.00	Cheques/P.O. payable to:
VDU Screen	**KADOR, Unit 4**
(Colour/Mono) £14.50	**Pontcynon Industrial Estate,**
Mouse Mat £5.95	**Abercynon**
Dust Covers	**Mid. Glamorgan CF45 4EP**
(Colour/Mono) £7.50	**Tel: 0443 740281**
Plonker Box £2.30	
Dexette Copy Holder £6.50	
Surge Protectors £12.00	
'Cheer Up' Mug £3.75	

kador

ADVERTISING IN BEEBUG

For advertising details, please contact Mike Williams on (0727) 40303
or write to:
Dolphin Place, Holywell Hill, St Albans Herts.AL1 1EX



This month Robin Burton looks at the vexed subject of just how compatible PC software is with the 512 co-processor.

One of the most common questions that 512 users ask BEEBUG is "How can I tell if this package will run?" or "Why doesn't my new version of Lotus 1-2-3 run, when the old version did?". There is no general answer to these questions, but we will show here some of the reasons why not all DOS software will run on the 512.

The first question is "Do you have DOS+ version 2.1?" If not, and you have problems, this is step one. DOS+ 2.1 fixes all known bugs from earlier versions, and provides new facilities too, including a utility needed for running 'pop-ups' like Sidekick. STL PC+ users should note that 2.1 also has a modified memory map to cater for this too.

Remember, if you seek help from your Acorn dealer, you can expect a fairly short answer if you are not using the current versions of the software. Keep a copy of your old DOS+ version though, because a few programs will not run with 2.1, but do with 1.2!

Compatibility is, to say the least, a large subject, but all the problems finally boil down to the fact that the 512 is not a PC clone. Some areas can be identified, and while there's no magical solution, at least some of the time wasted trying unsuitable software can be avoided by asking the right questions. There are also one or two tricks which might help and we'll look at these too.

PROBLEM AREAS

Strictly, problems can be due to either the 512 hardware, or its DOS+ operating system, but really the two cannot be separated. A complete and comprehensive compatibility list would be the perfect answer, but this isn't practical. There is no complete list, and it would fill several issues of BEEBUG if there was.

A better approach is to understand why packages may or may not run, and to be aware of pointers which apply to selecting software, the aim being to improve the chances of getting it right.

The simplest question is "How much memory does a package need?" For some it's 640k (standard for many PCs), so unless you have expanded your 512, these are out. Check this first. A second point is that DOS+ takes about 90K more memory for workspace than MS-DOS, so even if software is suitable for a 512K MS-DOS machine, do not assume that it is bound to work on the 512.

Next let's take a look at the hardware. It is the easiest area to deal with in many ways, not least because it is a known quantity, and in any case there is little you can do about it. Not surprisingly, to talk about the hardware we must also involve the operating system.

THE DOS ENVIRONMENT

It helps to know that DOS interfaces with applications by means of a system of fixed memory locations and associated operating system calls. These provide access to the software functions and the hardware too.

You are probably familiar with the various types of OS call in the BBC micro, but in DOS they are slightly different. In Acorn terms the nearest equivalent to a DOS call could perhaps be described as a delayed action OSWORD call, because they do not happen immediately on demand.

As you may know, some DOS systems can run more than one application simultaneously, or can split processing between foreground and background tasks (this is where the 'SLICE' command is used). The point is that DOS cannot be called on demand as the BBC micro's OS can, because no program can assume that it has the machine to itself. In consequence programs do not directly control the OS or the processor.

Instead, ten times every second, DOS polls all the programs that are running and examines every one of these defined memory locations to see if any program requires operating system attention. If so DOS carries out the required action before returning control to the user programs. System calls implemented in this way are called 'interrupts', but should not be confused with hardware interrupts.

An application should, in theory, use interrupts to read or write data when connecting to the outside world. In this context the outside world includes all peripherals attached to the machine, and remember that in DOS both the screen and the keyboard are treated as peripherals.

The object of all this is that DOS should always look the same to every user program, and vice versa, regardless of the micro on which it is run. In addition, enhancements can be included in later versions of DOS without disturbing any of the existing facilities. The result is a fixed, standard program environment, which is where the high degree of applications compatibility comes from.

THE HARDWARE INTERFACE

The only remaining question is "How can DOS deal with the hardware, so that it can be installed in different micros, some with slightly, or as in the case of the 512, very different architecture?"

Just as there is a standard protocol for applications wishing to talk to DOS, there is a standard for DOS when talking to hardware. In fact communication between DOS and the hardware is indirect and depends on the 'Basic Input Output System' (BIOS), which is a separate DOS module, customised for each different machine by the hardware supplier. In this way DOS itself need never be altered to cater for hardware peculiarities. Only the BIOS is ever changed, and because DOS provides an interface between the BIOS and the user's program, the operation of the latter should be unaffected.

Unfortunately it is not a perfect world, and just as in the BBC micro, while 'legal' software uses standard calls, not all software obeys the rules.

As an example, you probably know that in the BBC Micro the WD1770 or WD1772 chip controls the disc hardware. You probably also know that this chip can be directly 'programmed' so as to by-pass the disc filing system to read or write non-standard disc formats.

Similarly, in a true PC there are various chips or 'plug in' cards, each dedicated to a particular function, each of which can be be programmed either by using the 'legal' interrupts, or by 'illegal' (but generally faster) direct coding, again just like the BBC Micro.

PROBLEM DEFINED

Now to the crux of the problem. Contrast a true PC with the 512, where every external device is on the other side of the Tube. This means that only software using legal calls has even a chance of working. The real devices are simply not there. In consequence, software which attempts to read or write hardware devices directly will fail. (There's just one exception, and we will come to that in a moment.)

Even with legal calls, correct functioning depends on the ability of the 512's DOS+ and BIOS (actually called the XIOS by Acorn) to suitably translate or amend these calls into something the BBC micro on the other side of the tube will understand.

The job of DOS+ in the 512, therefore, is to translate legal DOS+ interrupts into calls across the Tube, but it can only do this for operations for which the BBC micro has both the appropriate hardware and a suitable software routine. Having finally reached the I/O processor, calls can then be actioned. This is performed by some extra code that the 512 pokes into the I/O processor's memory.

In short, to operate correctly DOS+ programs must only use legal interrupts which, ultimately, the BBC micro must be able to service. This is what determines whether DOS+ software runs or not.

Clearly discs and printers can be dealt with reasonably, and suitable coding can be provided for elementary screen and keyboard handling, but software that tries to read or

write hardware devices directly (or even legally for non-existent devices) must fail. This is why, for example, applications written for an Extended Graphics Adaptor (EGA) or a Hercules graphics card will not work. Even if DOS+ accepted the call, what could it do with it?

This gives the first compatibility pointer. The software you examine must be CGA compatible. Anything that demands an EGA, a Hercules or for that matter any other PC extension board will not work.

DOS DIFFERENCES

Now let's consider DOS in general, rather than the 512 or the BBC micro. Necessarily this is a simplification, but it will put things in perspective. The first point is that there are several members of the DOS family. The main ones are PC-DOS, used in IBM PCs, MS-DOS, produced by Microsoft and used by many clones, and DOS+, by Digital Research, a version of which is used by the 512 amongst others.

These are all similar enough to be obviously from the same family, yet at the same time they are different enough to cause compatibility problems with some applications.

A further complication, quite apart from the different implementations of DOS, is that for each type there may be several versions, hence the joke that not all IBM PCs are PC compatible. It depends on the version of DOS they use and the hardware devices catered for. As hardware has developed, DOS has been extended to cater for the changes.

Typically, applications written for a later versions of DOS will probably not function on an earlier version, even of the same type. Fortunately, the reverse is less common, and old software runs on later systems, more often than not (but not always).

512 SPECIFICS

DOS+ supplied with the 512 is compatible with all legally written software which runs with MS-DOS version 2.1, PC-DOS version 2.1, or CP/M(86) (the forerunner of DOS+).

This gives two more questions that you must ask when looking for new software. What type and what version of DOS is it compatible with? If it is not one of the above, you will probably have trouble.

Even when you are sure that DOS and the hardware is right, further confusion arises because some packages 'almost' run, but not quite, or most features work, but not all.

INITIAL SET-UP

Many DOS packages need installing, and have a special program for this purpose, but sometimes this is where the first problem occurs. Bear in mind that some packages will run on the 512, but cannot be installed on it. This can be because of initial defaults, or because 640k is needed for installation, though not necessarily when running.

The solution is to borrow a PC or clone to configure the system for the 512. Often the working system produced by this method can be used quite successfully.

Now to configuring itself. If there is a screen option for mono, choose it; if there isn't, select CGA. If there is a memory size option, select the one that suits the free memory given by 'BACKG'. Remember to allow for background printing, the alarm or the memory disc if you use them.

TRYING NEW SOFTWARE

When you run your new program, if things are still not right, try entering 'COMMAND' before loading the application. This re-loads 'COMMAND.COM', but the result is not quite the same as you get from booting the system. For some packages this seems to cure the problem.

Finally, when buying software, talk to people who should be able to help, like your Acorn dealer, but above all explain the situation. Any dealer worth his salt will understand, and allow a reasonable trial with an agreed refund or exchange if the package won't run. If he is not prepared to do this, the only advice is to take your money to someone who will.

THE NEW INTER-BASE REVIEWED

Lance Allison assesses Computer Concepts' InterBase, now re-released in a new and updated version.

Product	InterBase
Supplier	Computer Concepts Ltd Gaddesden Place, Hemel Hempstead, Herts HP2 6EX. Tel. (0442) 63933
Price	£67.85 inc. VAT and p&p

In the Summer of 1987 Computer Concepts released the final ROM in their Inter-Series. It was far more than just a simple database, incorporating a sophisticated programming language. Unfortunately this product was found to be somewhat less than bug-free, and its potential was limited as a result. At last, a year later, Computer Concepts has released a new version of the software that not only fixes these bugs, but introduces some very impressive new features.

After a year of waiting, the new version of InterBase has finally arrived. The package looks much the same as the first version. The box contains the InterBase ROM, a keystrip, an example disc, and a manual. The most obvious difference between the two versions is the size of the manual. It has grown to a large 250 pages, twice the size of the original.

Surprisingly enough the disc is only supplied in 80 track, single sided, DFS format. If this is not suitable for your system Computer Concepts recommends that you send it back for replacement. It would seem a good idea to state what system you have when ordering! InterBase works with practically all disc systems available for the BBC micro including the ADFS and even Solidisk's filing systems.

The ROM must be installed in one of your sideways ROM sockets, or in a cartridge if you have a Master. The ROM is supplied on a carrier board in usual Computer Concepts style. When the ROM has been installed the package is called up with the the *IBASE command. This will take you straight into the Card Index program.

At this point it is worth explaining the concept behind the InterBase programming language. InterBase is not just a standard database; it is a language with which to write a database system easily and efficiently just like DBase III for the PC. But if Computer Concepts were to market this programming language on its own, it would only be of interest to programmers. For this reason Computer Concepts has written a powerful database program (called the Card Index), which is supplied along with the language. It is this program that you enter upon typing *IBASE.

The Card Index presents you with a simple menu incorporating all of the options that you would expect to find in a good database. The menu is self explanatory and easy to follow. Databases may be created, edited, and updated at the touch of a key. More sophisticated utilities such as index creation and condition setting are also available and, unlike the previous version, appear to work faultlessly.

The beauty of this simple program is that you are totally in control of the record layout. You may select any screen mode for display, and may place fields on the screen in any order and in any position. At this level InterBase is comparable to most of the popular competitors on the market. Inevitably you will find limitations for your particular application and will want to expand the system. It is at this point that you may well want to make use of InterBase's own programming language.

Entry into the programming language is simply by selecting the 'Program menu' option in the Card Index menu. This will take you into a

menu which resembles that of a word processor such as Wordwise. This is, in fact, not far from the truth. A word processor is supplied in order to write and edit InterBase programs. The usual options are available such as save and load text, search and replace, and edit text. The editor allows many programs to be stored in memory at the same time, and each may be edited as required. To enter a program, just press Escape and type away.

The language resembles BBC Basic in many ways but offers more structuring without the use of line numbers. In addition to all of the usual Basic keywords, there is a whole host of extensions specifically designed for database manipulation. Entire databases may be created with single statements, and records may be saved and retrieved simply and quickly. Another major extension to the language is its handling of data structures. Basic is limited in that the elements of an array must all be of the same type. In InterBase, multi-dimensional arrays may contain a mixture of strings, integers, reals and dates (all in the same array). What is more, the whole array could be written to disc with a single command!

Possibly the most important improvement over the first version of InterBase is the ability to 'extend' the card index program. It is now easy to add another option to the Card Index menu to select your own utility. Control may then be returned to the Card Index upon completion.

COMPUTER CONCEPTS' ROM-LINK

As most people will be aware, Computer Concepts have devised the so-called ROM-LINK system with which all their packages may communicate. If you plan to use this system, InterBase is far more than just another member; it ties the whole system together because of its programming ability. It could be used to allow a database to perform a mail merge with InterWord, or even to display information using InterChart. The possibilities for a complete interactive data system are endless.

Although programs are written using InterBase, they may in fact be run from within any of the other ROM-LINK packages. There is a nice example in the manual demonstrating how a short InterBase program can be used to calculate thirty random numbers for InterChart so that they can be displayed graphically. Although InterBase is not being used for its database abilities it shows how the system slots together. The manual is full of useful demonstrations and exercises of this kind.

DOCUMENTATION

Back in August 1987 Peter Rochford reviewed the first version of InterBase for BEEBUG (Vol.6 No.4). His principal criticism was the standard of documentation. I am glad to say that it has been improved substantially. More emphasis has been put on tutorials, and many more examples have been included. This is reflected in the size of the manual which has grown from one hundred to two hundred and fifty leaves. This is not to say that the documentation is perfect; there is still room for improvement.

CONCLUSION

If you use any of the Inter Series ROMs for business or pleasure, InterBase is a must. Educational institutions may find that the InterBase language is a good introduction to concepts to be found in more expensive commercial packages such as DBase III.

Freelance programmers may use InterBase to write specialist database applications. Resulting programs may be blown into ROM and used completely independent of the Card Index. The only limitation is that an InterBase ROM must be installed in the machine in which the program is to run.

EXISTING INTERBASE OWNERS

Computer Concepts are offering a free upgrade to all customers who have purchased the first version of InterBase. Contact Computer Concepts directly for further information.

B

THE BEEBUG SUPER-SQUEEZE

Are your programs tight on bytes? David Spencer can help with his latest utility.

It is often necessary with long Basic programs to reduce their length as much as possible so that they will fit into the available memory. This can be done in a number of ways, the obvious two being the removal of comments and unnecessary spaces. However, such crunching of a program by hand is time-consuming and prone to error. This is where The BEEBUG Super-Squeeze steps in. It crunches a program as much as possible by removing REMs, blank lines, spaces, THENs, LETs, shortening variable names, and concatenating lines.

USING THE SUPER-SQUEEZE

The Beebug Super-Squeeze is in the form of a ROM image which must be loaded into sideways RAM. To assemble the machine code, enter the listing given here, save and run it. This will create the ROM image and save it to disc using the name 'SQZOBJ'. The method for loading Super-Squeeze depends on the type of sideways RAM you are using. For a Master 128, Compact or B+128K, you can use:

```
*SRLOAD SQZOBJ 8000 ZQ
```

For other types of sideways RAM you should refer to their instruction books for loading details. Once the ROM image is loaded you should press Ctrl-Break to initialise it. It is important not to write-protect the RAM bank containing Super-Squeeze, because the spare memory is used as workspace.

To crunch a Basic program, it must first be loaded into memory. Then type *SQUEEZE (minimum abbreviation *SQ.), and Super-Squeeze will do its best to shorten the program to the minimum possible length. This may take a few minutes with a long program, and Escape can be used to stop the crunching at any point. As the crunching proceeds, dots are printed to give an idea of the progress.

Once a program has been crunched using Super-Squeeze, you should not attempt to edit it at all. This is because the Basic line parser gets confused by the lack of spaces around keywords and fails to tokenise the program correctly. You should always keep a safe 'source' version of any program compacted in this way.

Another possible problem is with the use of the function EVAL to evaluate an expression. Within the argument to the EVAL function it is possible to include variable names within quotes. For example:

```
PRINT EVAL("prefix$+"+A$)
```

Super-Squeeze will not recognise the text within the quotes as referring to a variable, but will shorten the name 'prefix$' elsewhere in the program. This results in a 'No such variable' error when the EVAL is executed.

HOW IT WORKS

We do not have the space here to explain the complexities of Super-Squeeze. Suffice it to say that two passes of the program are made. On the first pass a list of all variable names and line references is made. On the second pass the actual crunching is performed.

```
 10 REM Program Super-Squeeze
 20 REM Version B1.00
 30 REM Author  David Spencer
 40 REM BEEBUG  October 1988
 50 REM Program subject to copyright
 60 :
100 page=&18:lptr=&51:lnos=&53
110 vars=&55:temp=&57:temp2=&58
120 aflg=&59:lnum=&5A:ltemp=&5D
130 vend=&5F:bst=&61:newv=&63
140 pend=&66:blen=&6A:glen=&6C
150 bptr=&6E:oeflg=&70:tflg=&71
160 sflg=&72:lboff=&73:eflg=&74
170 oeflg=&75:boff=&76:ytemp=&77
180 quof=&78:lin=&79:yback=&7A
190 vtype=&7B:hexflg=&7C
200 DIM code 3000
210 FOR pass=4 TO 7 STEP 3
220 P%=&8000:O%=code
230 [OPT pass:EQUB 0:EQUB 0:EQUB 0
240 JMP service:EQUB &82
250 EQUB copy AND &FF:EQUB 1
260 EQUS "BEEBUG Super-Squeeze"
270 .copy EQUB 0
280 EQUS "(C) BEEBUG 1988":EQUB 0
290 .service:CMP #4:BEQ comm:RTS
```

```
300 .comm TYA:PHA:LDX #0
310 .comm2 LDA (&F2),Y:AND #&DF
320 CMP ourc,X:BNE comm4:INY:INX
330 CPX #7:BNE comm2
340 .comm3 LDA (&F2),Y:INY:CMP #ASC" "
350 BEQ comm3:BCC ours:BCS comm5
360 .comm4 LDA (&F2),Y:INY:CMP #ASC"."
370 BEQ comm3:.comm5 PLA:TAY:LDX &F4
380 LDA #4:RTS:.comm6 JSR &FFE7:PLA
390 TAY:LDX &F4:LDA #0:RTS
400 .ourc EQUS "SQUEEZE"
410 .ours LDA #&FF:STA lnos:STA end
420 LDA #&BF:STA lnos+1:LDA page
430 STA lptr+1:LDA #end MOD &100
440 STA vend:LDA #end DIV &100
450 STA vend+1:LDY #1:STY lptr:DEY
460 LDA (lptr),Y:BMI comm6:STY aflg
470 .ploop LDY #3:.ploop2:LDA (lptr),Y
480 CMP #&F4:BNE ploop3:JMP pcr
490 .ploop3 CMP #13:BEQ pcrj
500 CMP #ASC"""":BNE pnq:.qloop INY
510 LDA (lptr),Y:CMP #13:BNE ql2
520 .pcrj JMP pcr:.ql2 CMP #ASC""""
530 BNE qloop:INY:BNE ploop2
540 .pnq CMP #ASC"&":BNE namp
550 .ampsk INY:LDA (lptr),Y
560 CMP #ASC"0":BCC ploop2
570 CMP #ASC"9"+1:BCC ampsk
580 CMP #ASC"A":BCC ploop2
590 CMP #ASC"F"+1:BCC ampsk:BCS ploop2
600 .namp CMP #ASC"[":BNE nason
610 LDA #&FF:.nas STA aflg:INY
620 BNE ploop2:.nason CMP #ASC"]"
630 BNE nasof:LDA #0:BEQ nas
640 .nasof CMP #&A4:BEQ pf:CMP #&F2
650 BNE npf:.pf STA vbuff:LDX #1
660 .pf2 INY:LDA (lptr),Y:JSR chkdig
670 BCS pf3:JSR vschk:BCC pf4
680 .pf3 STA vbuff,X:INX:BNE pf2
690 .pf4 LDA #0:STA vbuff,X:JSR vadd2
700 JMP ploop2
710 .npf CMP #&8D:BNE nlno:INY
720 JSR decode:STY temp:JSR nocor
730 LDY #0:STA (lnos),Y:JSR dlnos
740 TXA:STA (lnos),Y:JSR dlnos
750 LDY temp:.plj JMP ploop2
760 .nlno INY:BIT aflg:BPL ninas
770 CMP #ASC".":BNE plj
780 .nlno2 LDA (lptr),Y:INY
790 CMP #ASC" ":BEQ nlno2
800 .ninas JSR vschk:BCC plj:JSR vadd
810 JMP ploop2
820 .pcr BIT &FF:BPL pcr2:JMP comm6
830 .pcr2 JSR nxtlin:LDY #0
840 LDA (lptr),Y:BMI pcr3:JMP ploop
850 .pcr3 LDA lptr:CLC:ADC #1:STA pend
860 LDA lptr+1:ADC #0:STA pend+1
870 JMP pld
880 .decode LDA (lptr),Y:ASL A:ASL A
890 STA temp:AND #&C0:INY:EOR (lptr),Y
900 TAX:LDA temp:ASL A:ASL A:INY
910 EOR (lptr),Y:INY:RTS
920 .dlnos LDA lnos:BNE dlnos2
930 DEC lnos+1:.dlnos2 DEC lnos
940 LDA lnos:CMP vend:BNE dlnos3
950 LDA lnos+1:CMP vend+1:BNE dlnos3
960 JMP vnroom:.dlnos3 RTS
970 .vadd JSR vpull:.vadd1 LDA vbuff
980 CMP #ASC"%":BNE vadd2:LDA vbuff+2
990 BNE vadd2:LDA vbuff+1:CMP #ASC"A"
1000 BCC vadd2:CMP #ASC"Z"+1:BCS vadd2
1010 RTS:.vadd2 JSR vfind:BCC vput:RTS
1020 .vput STY temp:LDY #0
1030 .vput2 LDA vbuff,Y:STA (vars),Y
1040 INY:CMP #0:BNE vput2:LDA #&FF
1050 STA (vars),Y:TYA:CLC:ADC vars:PHA
1060 STA vend:LDA vars+1:ADC #0
1070 STA vend+1:CMP lnos+1:BCC vsok
1080 BNE vnroom:PLA:CMP lnos:BCC vsok2
1090 .vnroom LDA #0:STA &100:STA &101
1100 TAY:.eloop LDA verr,Y:STA &102,Y
1110 INY:CMP #0:BNE eloop:JMP &100
1120 .verr EQUS "No room":EQUB 0
1130 .vsok PLA:.vsok2 LDY temp:RTS
1140 .vfind STY temp:LDA #end MOD &100
1150 STA vars:LDA #end DIV &100
1160 STA vars+1:LDA #ASC"A":STA newv
1170 LDA #0:STA newv+1:STA newv+2
1180 LDA vbuff:CMP #ASC"%":BNE vlo
1190 LDA #ASC"_":STA newv
1200 .vlo LDY #0:LDA (vars),Y:STA vtype
1210 CMP #&FF:BEQ vfind2
1220 .vloop LDA (vars),Y:CMP vbuff,Y
1230 BNE vloop2:INY:CMP #0:BNE vloop
1240 LDY temp:SEC:RTS
1250 .vloop2 LDA (vars),Y:PHP:INY:PLP
1260 BNE vloop2:LDA vtype:CMP vbuff
1270 BNE vloop3:JSR incvar:.vloop3 CLC
1280 TYA:ADC vars:STA vars:BCC vlo
1290 INC vars+1:BCS vlo:.vfind2 CLC
1300 LDY temp:RTS
1310 .vschk CMP #ASC"A":BCC vschk3
1320 CMP #ASC"z"+1:BCS vschk3
1330 CMP #ASC"_":BCS vschk2:CMP #ASC"["
1340 BCS vschk3:.vschk2 SEC:RTS
1350 .vschk3 CLC:RTS
1360 .chknul LDY #3:.chkn2 LDA (lptr),Y
1370 INY:CMP #ASC" ":BEQ chkn2
1380 CMP #ASC":":BEQ chkn2:CMP #13
1390 BEQ chkn3:CMP #&F4:BEQ chkn3
1400 CMP #&8B:.chkn3 RTS
1410 .encode PHA:ORA #&40:STA lnum+2
1420 TXA:AND #&3F:ORA #&40:STA lnum+1
1430 TXA:AND #&C0:STA temp2:PLA
```

```
1440 AND #&C0:LSR A:LSR A:ORA temp2
1450 LSR A:LSR A:EOR #&54:STA lnum:RTS
1460 .nocor STA ltemp+1:STX ltemp
1470 LDA lptr:PHA:LDA lptr+1:PHA:LDA #1
1480 STA lptr:LDA page:STA lptr+1
1490 .noc LDY #0:LDA (lptr),Y
1500 BMI nocout:CMP ltemp+1:BNE nonxt
1510 INY:LDA (lptr),Y:CMP ltemp
1520 BEQ nofnd:.nonxt JSR nxtlin
1530 JMP noc:.nocout PLA:STA lptr+1:PLA
1540 STA lptr:.nocout2:LDA ltemp+1
1550 LDX ltemp:RTS:.nofnd JSR chknul
1560 BNE gofnd:JSR nxtlin:JMP nofnd
1570 .gofnd LDY #1:LDA (lptr),Y
1580 STA ltemp:TAX:DEY:LDA (lptr),Y
1590 STA ltemp+1:JSR encode:PLA
1600 STA lptr+1:PLA:STA lptr:LDY temp
1610 DEY:LDA lnum+2:STA (lptr),Y:DEY
1620 LDA lnum+1:STA (lptr),Y:DEY
1630 LDA lnum:STA (lptr),Y:BNE nocout2
1640 .nxtlin LDY #2:LDA (lptr),Y:CLC
1650 ADC lptr:STA lptr:BCC nxtlin2
1660 INC lptr+1:.nxtlin2 RTS
1670 .vpull STA vbuff+1:LDX #2
1680 .vpull2 LDA (lptr),Y:INY
1690 JSR chkdig:BCS vpull3:JSR vschk
1700 BCC vdn:.vpull3 STA vbuff,X:INX
1710 BNE vpull2:.vdn STY yback
1720 DEC yback:PHA:LDA #0:STA vbuff,X
1730 LDA #1:STA temp:PLA:CMP #ASC"%"
1740 BEQ vdn2:CMP #ASC"$":BNE vdn3
1750 .vdn2 STA temp:LDA (lptr),Y:INY
1760 .vdn3 CMP #ASC"(":BNE vdn4:INY
1770 LDA temp:ORA #&80:STA temp
1780 .vdn4 DEY:LDA temp:STA vbuff:RTS
1790 .vmatch JSR vpull:STY temp+1
1800 LDY yback:.vmatch2 JSR vfind
1810 BCC vmatch5:BIT hexflg
1820 BMI vmatch25:LDA newv:CMP #ASC"G"
1830 BCS vmatch25:JSR sadd
1840 .vmatch25 LDA newv:JSR badd
1850 LDA newv+1:BEQ vmatch3:JSR badd
1860 LDA newv+2:BEQ vmatch3:JSR badd
1870 .vmatch3 CPY temp+1:BEQ vmatch4
1880 LDA (lptr),Y:JSR badd:INY
1890 BNE vmatch3:.vmatch4 SEC:RTS
1900 .vmatch5:LDY #1:.vmatch6
1910 LDA vbuff,Y:BEQ vmatch7:JSR badd
1920 INY:BNE vmatch6:.vmatch7 LDA vbuff
1930 PHA:AND #&7F:CMP #1:BEQ vmatch8
1940 JSR badd:.vmatch8 PLA:BPL vmatch9
1950 LDA #ASC"(":JSR badd
1960 .vmatch9 LDY temp+1:CLC:RTS
1970 .incvar LDA newv:JSR vinc:STA newv
1980 BNE incvd:LDA #ASC"A":STA newv
1990 LDA newv+1:JSR vinc:STA newv+1
2000 BNE incvd:LDA #ASC"0":STA newv+1
```

```
2010 LDA newv+2:JSR vinc:STA newv+2
2020 .incvd RTS
2030 .vinc CLC:ADC #1:CMP #1:BNE vinc1
2040 LDA #&30:.vinc1 CMP #&3A:BNE vinc2
2050 LDA #&41:.vinc2 CMP #&5B:BNE vinc3
2060 LDA #&5F:.vinc3 CMP #&7B:RTS
2070 .shunt LDA lbuff:CMP #&80
2080 BNE shunt2:RTS:.shunt2 LDA #ASC"."
2090 JSR &FFEE:LDA pend:SEC:SBC lptr
2100 STA blen:LDA pend+1:SBC lptr+1
2110 STA blen+1:ORA blen:BEQ shuo
2120 LDA lptr:SEC:SBC bst:TAX
2130 LDA lptr+1:SBC bst+1:PHA:TXA:SEC
2140 SBC lbuff+2:STA glen:PLA:SBC #0
2150 STA glen+1:BCC shuup:ORA glen
2160 BEQ shuo:LDA lptr:STA temp:SEC
2170 SBC glen:STA bptr:LDA lptr+1
2180 STA temp+1:SBC glen+1:STA bptr+1
2190 LDY #0:.shudnl LDA (temp),Y
2200 STA (bptr),Y:INY:BNE shudnl2
2210 INC temp+1:INC bptr+1
2220 .shudnl2:JSR decb:BNE shudnl
2230 BEQ shuo:.shuup LDA pend:STA temp
2240 SEC:SBC glen:STA bptr:LDA pend+1
2250 STA temp+1:SBC glen+1:STA bptr+1
2260 LDY #0:.shuup2 LDA bptr:BNE shuup3
2270 DEC bptr+1:.shuup3 DEC bptr
2280 LDA temp:BNE shuup4:DEC temp+1
2290 .shuup4 DEC temp:LDA (temp),Y
2300 STA (bptr),Y:JSR decb:BNE shuup2
2310 .shuo LDY #0:.shuo2 LDA lbuff,Y
2320 STA (bst),Y:INY:CPY lbuff+2
2330 BNE shuo2:LDA lptr:SEC:SBC glen
2340 STA bst:STA lptr:LDA lptr+1
2350 SBC glen+1:STA bst+1:STA lptr+1
2360 LDA pend:SEC:SBC glen:STA pend
2370 LDA pend+1:SBC glen+1:STA pend+1
2380 RTS
2390 .decb LDA blen:BNE decb2
2400 DEC blen+1:.decb2 DEC blen
2410 LDA blen:ORA blen+1:RTS
2420 .toktab
2430 EQUB &D5:EQUB &FB:EQUB &D6
2440 EQUB &D7:EQUB &D9:EQUB &DE
2450 EQUB &DF:EQUB &E2:EQUB &E3
2460 EQUB &E4:EQUB &E5:EQUB &E6
2470 EQUB &D3:EQUB &E7:EQUB &E8
2480 EQUB &D2:EQUB &EA:EQUB &EB
2490 EQUB &EC:EQUB &ED:EQUB &EE
2500 EQUB &FF:EQUB &F1:EQUB &D0
2510 EQUB &CF:EQUB &F0:EQUB &F2
2520 EQUB &F3:EQUB &F7:EQUB &D4
2530 EQUB &FC:EQUB &D1:EQUB &FD
2540 EQUB &EF:EQUB &FE:EQUB 0
2550 .tokchk STY ytemp:LDY #0:TAX
2560 .tokchk2 TXA:CMP toktab,Y
2570 BEQ tokchk3:INY:LDA toktab,Y
```

```
2580 BNE tokchk2:LDA #&FF:.tokchk3 PHP
2590 LDY ytemp:PLP:RTS
2600 .badd LDX boff:STA sbuff,X:STA lin
2610 INC boff:RTS
2620 .sadd PHA:LDA #32:JSR badd:PLA:RTS
2630 .chkdig CMP #ASC"0":BCC chkdig3
2640 CMP #ASC"9"+1:BCS chkdig3:SEC:RTS
2650 .chkdig3 CLC:RTS
2660 .p1d LDY #1:STY lptr:STY bst
2670 LDA page:STA lptr+1:STA bst+1
2680 LDA #&80:STA lbuff:.chew LDY #0
2690 LDA (lptr),Y:BMI chout:BIT &FF
2700 BPL chew05:.chout JSR shunt
2710 JMP comm6:.chew05 STY oeflg
2720 STY tflg:JSR chknul:BNE chew1
2730 JSR nxtlin:JMP chew:.chew1 LDY #0
2740 STY lin:.chew2 LDA (lptr),Y
2750 STA sbuff,Y:INY:CPY #3:BNE chew2
2760 STY boff:DEY:.chew3 LDA #0
2770 STA sflg:STA eflg:LDA #&FF
2780 STA hexflg:.chew4 INY
2790 .chew45 LDA hexflg:BMI chew46
2800 DEC hexflg:.chew46 LDA (lptr),Y
2810 CMP #13:BEQ dnj2
2820 CMP #ASC":":BNE chew5:LDA lin
2830 BEQ chew3:CMP #ASC":":BEQ chew3
2840 LDA #ASC":":JSR badd:JMP chew3
2850 .chew5 CMP #ASC" ":BEQ chew4
2860 CMP #&E9:BEQ chew4:CMP #ASC"*"
2870 BNE chew6:JSR badd:BIT eflg
2880 BMI chew4:DEC sflg:.osc INY
2890 LDA (lptr),Y:CMP #13:BEQ dnj2
2900 JSR badd:JMP osc:.dnj2 JMP chewdn
2910 .chew6 CMP #ASC",":BNE chew7
2920 JSR badd:.chewj JMP chew4
2930 .chew7 CMP #&DC:BNE chew8
2940 JSR badd:LDA #0:STA quof:.data INY
2950 LDA (lptr),Y:CMP #ASC" ":BEQ data
2960 DEY:.data2 INY:LDA (lptr),Y
2970 CMP #13:BEQ dnj:JSR badd
2980 CMP #ASC"""":BNE data3:LDA quof
2990 EOR #&80:STA quof:JMP data2
3000 .data3 BIT quof:BMI data2
3010 CMP #ASC",":BNE data2:JMP data
3020 .chew8 CMP #ASC"""":BNE chew9
3030 JSR badd:.qte INY:LDA (lptr),Y
3040 JSR badd:CMP #13:.dnj BEQ dnj2
3050 CMP #ASC"""":BNE qte:BEQ chewj
3060 .chew9 CMP #ASC"&":BNE chew10
3070 .hexx JSR badd:INY:LDA (lptr),Y
3080 JSR chkdig:BCS hexx:CMP #ASC"A"
3090 BCC hexx2:CMP #ASC"G":BCC hexx
3100 .hexx2 DEY:.hexx3:INY:LDA (lptr),Y
3110 CMP #ASC" ":BEQ hexx3:CMP #ASC"A"
3120 BCC hexx4:CMP #ASC"G":BCS hexx4
3130 JSR sadd:.hexx4 LDA #1:STA hexflg
3140 JMP chew45:.chew10:BIT eflg
```

```
3150 BMI chew11:CMP #ASC".":BNE chew11
3160 JSR badd:JMP chew4
3170 .chew11 CMP #ASC".":BEQ chew12
3180 JSR chkdig:BCC chew13
3190 .chew12 JSR badd:INY:LDA (lptr),Y
3200 CMP #ASC".":BEQ chew12:JSR chkdig
3210 BCS chew12:DEY:LDA #&FF:STA eflg
3220 .chew121 INY:LDA (lptr),Y
3230 CMP #ASC" ":BEQ chew121:JSR chkdig
3240 BCC chew122:JSR sadd
3250 .chew122 JMP chew45
3260 .chew13 CMP #&8D:BNE chew14
3270 JSR badd:LDA #3:STA temp+1
3280 .lnch INY:LDA (lptr),Y:JSR badd
3290 DEC temp+1:BNE lnch
3300 .chewj2 JMP chew4
3310 .chew14 CMP #ASC"A":BCS chew15
3320 JSR badd:LDA #&FF:STA eflg
3330 BNE chewj2:.chew15 CMP #ASC"\"
3340 BNE chew16:.ccsk INY:LDA (lptr),Y
3350 CMP #13:BNE ccsk2:JMP chewdn
3360 .ccsk2 CMP #ASC":":BNE ccsk
3370 .chewjj JMP chew45
3380 .chew16 JSR vschk:BCC chew17:INY
3390 JSR vmatch:BCS itwas:LDA vbuff
3400 CMP #1:BEQ vsep5:.itwas LDA #&FF
3410 STA eflg:DEY:LDA (lptr),Y
3420 CMP #ASC"$":BEQ vnsep:CMP #ASC"("
3430 BNE vsep:.vnsep JMP chew4
3440 .vsep STA temp:INY:LDA (lptr),Y
3450 CMP #ASC" ":BEQ vsep2:JMP chew45
3460 .vsep2 INY:LDA (lptr),Y
3470 CMP #ASC" ":BEQ vsep2:CMP #ASC"?"
3480 BEQ vsep4:CMP #ASC"!":BEQ vsep4
3490 LDA temp:JSR chkdig:BCS vsep3
3500 JSR vschk:BCS vsep3:JMP chew45
3510 .vsep3 LDA (lptr),Y:CMP #ASC"."
3520 BEQ vsep4:JSR chkdig:BCS vsep4
3530 JSR vschk:BCC vsep5
3540 .vsep4 JSR sadd:.vsep5 JMP chew45
3550 .chew17 CMP #&8C:BNE chew18
3560 .th1 INY:LDA (lptr),Y:CMP #ASC" "
3570 BEQ th1:CMP #&CF:BCS th12
3580 CMP #&8D:BNE th13:.th14 LDA #&8C
3590 JSR badd:JMP th12
3600 .th13 CMP #ASC"*":BEQ th14
3610 JSR sadd:.th12 LDA #0:STA eflg
3620 JMP chew45:.chew18 CMP #&A4
3630 BEQ fnnn:CMP #&F2:BNE chew19
3640 .fnnn JSR badd:STA vbuff:LDX #1
3650 .fnnn2 INY:LDA (lptr),Y:JSR chkdig
3660 BCS fnnn3:JSR vschk:BCC fnnn4
3670 .fnnn3 STA vbuff,X:INX:BNE fnnn2
3680 .fnnn4 LDA #0:STA vbuff,X
3690 JSR vfind:LDA newv:JSR badd
3700 LDA newv+1:BEQ fnnn5:JSR badd
```

Continued on page 64

USING ASSEMBLER

PART 3

This month Lee Calcraft discusses vectors, and the implementation of a dual-window system.

Vectors provide the BBC micro programmer with a very useful tool in that they allow him to "bolt on" pieces of code to the machine's resident firmware. For example, by using vectors it is possible to add new Basic commands or new star commands to the BBC micro's repertoire, or to implement such things as pop-up calculators.

This is made possible because a number of vital entry points into the machine's firmware are *vectored*. This means that when commonly used routines such as OSRDCH (operating system read character) are called, whether it be by the user, by Basic, or by the operating system itself, the program does not immediately jump to the OSRDCH code in ROM and begin execution. It first looks at a fixed location in RAM called the OSRDCH vector, and jumps to the address held by this vector. Normally this address will be that of the corresponding ROM routine. This is because when the machine powers up, or when Break is pressed, the operating system installs the addresses of all default routines at the appropriate vectors. Each vector consists of two bytes of RAM, and holds the 16 bit address of its service routine low byte first.

It is thus a simple matter to replace any vector with an address which points to a user-supplied routine. In most cases the new code supplied by the user will not *replace* the resident code, but will supplement it. Thus for example, if we were installing a pop-up calculator, we might wish to add a special routine to OSRDCH to check for a particular key code (e.g. Ctrl-P) in the keyboard buffer. If the ASCII code for Ctrl-P (16) was detected, then the calculator would be invoked. But if not, we would want the keypress to be processed normally. To ensure that this occurs, we simply need to save the original contents of the read-character vector at &210 and &211, and jump to the address formed by those contents after we have made our check for Ctrl-P. Broadly speaking the same process occurs when using any of the machine's vectors, and a list of some of the more common ones is given in table 1.

200	User vector
202	Break vector
204	IRQ vector
208	Command line interpreter
20E	OSWRCH write character vector
210	OSRDCH read character vector
212	OSFILE filing system vector
214	OSARGS filing system vector
216	OSBGET filing system vector
218	OSBPUT filing system vector
21A	OSGBPB filing system vector

Table 1. Commonly used vectors

A PRACTICAL EXAMPLE

In order to illustrate the way in which vectors can be used, we will take a very simple example. The object of the exercise will be to produce a beep whenever a key is pressed. The VDU7 beep provides a very useful way of testing new routines since it is both easy to implement, and easy to detect when activated.

The program in listing 1 uses the read-character vector in this way. If you type it in, and run it, you will find that the computer will operate more or less as normal, except that every time a character is read from the current input stream (e.g. after a keypress), a beep will be sounded. To terminate the effect, ignore the beeps, and type:

```
CALL reset
```

As you can see from the listing, there are three separate routines (*setup*, *newcode* and *reset*).

Most programs which make use of vectors can be broken down into three such parts, and it will be instructive to look at each in turn.

Listing 1

```
 10 REM Read chr vector
 20 REM Author  Lee Calcraft
 30 REM Version B 0.2
 40 :
 50 oswrch=&FFEE
 60 vector=&210:REM read chr vector
 70 newvec=&70 :REM Temp storage
 80 MODE7
 90 :
100 FOR pass=0 TO 1
110 P%=&900
120 [
130 OPT pass*3
140 .setup     \Set up new vector
150            \to point to new code
160 LDA vector:STA newvec
170 LDA vector+1:STA newvec+1
180 LDA #newcode MOD &100
190 STA vector
200 LDA #newcode DIV &100
210 STA vector+1
220 RTS
230 :
240 .newcode  \Produce beep
250 PHA
260 LDA #7
270 JSR oswrch
280 PLA
290 JMP (newvec)
300 :
310 .reset     \Reset vector
320 LDA newvec:STA vector
330 LDA newvec+1:STA vector+1
340 RTS
350 ]
360 NEXT
370 CALL setup
```

The purpose of *setup* is to store the original contents of the read-character vector (situated at locations &210 and &211), and to replace these two bytes with the address of our new routine, which is situated at *newcode*. Note here, the way in which MOD and DIV are used to calculate the low and high bytes of *newcode*. The hash (#) in lines 180 and 200 is also worth noting. This is essential because although *newcode* is an address, we are loading its low and high byte components as immediate data.

The routine *setup* is called just once immediately after assembly. From then on, every time that the computer reads a character from the input stream, it automatically accesses *newcode*. In this example *newcode* does very little. It first stacks the accumulator, then loads it with the value 7, and performs a jump to OSWRCH to send the character 7 to the VDU, and thus produce a beep. Finally it unstacks the accumulator and jumps to the address held at *newvec* and *newvec+1*. This is our own new vector containing the original contents of &210 and &211. This means that the normal read-character routine will now be executed. And since we have preserved the value held in the accumulator by stacking and unstacking it, the correct character will be read by OSRDCH.

Finally we come to *reset*. This routine (activated from Basic by CALL reset) restores the original contents to the read-character vector, so that *newcode* will no longer be accessed at every keypress. To reactivate the program you can always use *CALL setup* again. But there is just one thing to avoid: calling *setup* when the new vector is already in place. So when debugging the program you will need to clear the new vector each time before you run the Basic program (since the Basic program also calls *setup*). This can be done either by pressing Break, or by calling *reset*.

IMPLEMENTING DUAL WINDOWS

To give a more practical illustration of the use of vectors, listing 2 implements a dual-window system. When it is installed, pressing Ctrl-P from the keyboard will toggle between two text windows situated side by side on the screen. You can use them to catalogue a disc while keeping a program listing on screen, or to compare two parts of a program, etc. Once enabled, the routine not only operates from immediate mode Basic, but at any time that keypresses are read from the keyboard buffer. For example, during Basic's INPUT, GET or INKEY (but not when using negative INKEY),

or from any machine code character input routine which uses OSRDCH.

To put the program through its paces, type it in, and save it away. When it is run it will install the new vector. As a result the computer will behave normally until Ctrl-P is pressed. At this point mode 3 will be engaged, and a text window set up on the left-hand side of the screen. Now the computer can again be used normally, though all text will appear within this left-hand window. To switch to the right-hand window, simply press Ctrl-P again. Each time that a window is reselected, the cursor will be reinstated to the position which it held when the window was last used. To put things back to normal, use Ctrl-Shift-P

If you wish to save the code, use:

```
*SAVE windows 900 +FF
```

Typing *windows at any time will then install the routine and make it active. But take care to do this only if you are not already using the windows.

HOW IT WORKS

The program has many elements in common with that in listing 1. The *setup* routine is identical except that a few lines have been added to it to initialise three memory locations: one holding a flag giving the current state of the dual window system, and those used as a temporary store of the cursor position. The section of the program at *newcode* has also been extended. Here we first check to see whether the accumulator holds the value 16, indicating that Ctrl-P has been pressed. If not, we hastily exit. But if Ctrl-P is detected, we stack the X and Y registers, and perform a negative INKEY (OSBYTE &81) to check for the Shift key. If Shift is detected, the reset routine is entered. This resets the vectors to normal, cancels the windows (with VDU26), emits a beep and returns to Basic.

If Shift is not detected, a jump is made to the subroutine *thecode*. This contains the whole of the windowing routine. On entry to *thecode*, the state of the window flag is tested, and if it contains zero, indicating that the routine has been entered for the first time, it engages mode 3. Next the current position of the cursor is read by a call to OSBYTE &86, and stored in *tempx* and *tempy*. The window is then set, depending on the contents of the bottom bit of the window flag (tested by LSR A followed by BCC in lines 860 and 870). Finally the cursor is placed at the correct position for that window, the old cursor position stored for use next time, and the window flag adjusted accordingly.

Dual windows in mode 3

If you wish to modify the program to generate different sized windows, you can alter the data from line 1100 onwards. Clearly there is great scope for experimentation. You might replace the window routine for one which swapped the RAM used for the function keys on a model B (or character definitions). Or you could set up a system for entering Basic keywords at a two-key press. If you feel really ambitious you could even try writing a new Basic keyword. For help in all these matters, you will find *The Advanced User Guide* to be quite indispensable.

Next month we will look into the associated topic of events and interrupts.

Listing 2

```
  10 REM Program Dual Windows
  20 REM Version B 1.0b
  30 REM Author  Lee Calcraft
  40 REM BEEBUG  October 1988
  50 REM Program Subject to Copyright
  60 :
 100 oswrch=&FFEE:osbyte=&FFF4
```

```
110 vector=&210:REM read chr vector
120 newvec=&70:flag=&72:
130 curx=&73:cury=&74
140 tempx=&75:tempy=&76
150 MODE3
160 :
170 FOR pass=0 TO 1
180 P%=&900
190 [
200 OPT pass*3
210 .setup    \Set up new vector
220 LDA vector:STA newvec
230 LDA vector+1:STA newvec+1
240 LDA #newcode MOD &100
250 STA vector
260 LDA #newcode DIV &100
270 STA vector+1
280 LDA #0
290 STA flag  \Initialise flag
300 STA curx:STA cury \and cursor
310 STA tempx:STA tempy
320 RTS
330 :
340 .newcode  \Check key pressed
350 PHP:PHA   \Stack status & acc
360 CMP #16   \Test for Ctrl P
370 BNE notmykey
380 TXA:PHA   \Stack X register
390 TYA:PHA   \Stack Y register
400 :
410 LDY #&FF  \Test for Shift
420 LDX #&FF
430 LDA #&81  \Perform inkey(-1)
440 JSR osbyte
450 :
460 CPX #&FF
470 BEQ reset \Shift so reset
480 JSR thecode\Call window code
490 :
500 .skipit
510 PLA:TAY   \Unstack Y register
520 PLA:TAX   \Unstack X register
530 .notmykey
540 PLA:PLP   \Unstack status & acc
550 JMP (newvec)\Resume OS routine
560 :
570 .reset    \Reset vector
580 LDA newvec:STA vector
590 LDA newvec+1:STA vector+1
600 LDA #26:JSR oswrch
610 LDA #7:JSR oswrch
620 JMP skipit
630 :
640 .thecode
650 LDA flag
660 BNE skipclear\Test flag
670 TAX
680 LDA #22:JSR oswrch\Mode 3
690 LDA #3:JSR oswrch
700 TXA
710 .skipclear
720 PHA       \Stack the flag
730 LDA #&86
740 JSR osbyte\Get cursor posn
750 STX tempx
760 STY tempy
770 PLA       \Unstack the flag
780 :
790 LDX #0    \Set up window
800 LSR A
810 BCC window
820 LDX #5
830 .window
840 LDY #5
850 .loop
860 LDA params,X
870 JSR oswrch
880 INX
890 DEY
900 BNE loop
910 :
920 LDA #&1F  \Set cursor
930 JSR oswrch
940 LDA curx
950 JSR oswrch
960 LDA cury
970 JSR oswrch
980 :
990 LDA tempx \Save old cursor
1000 STA curx
1010 LDA tempy
1020 STA cury
1030 :
1040 LDA flag  \Adjust flag
1050 ORA #&80
1060 EOR #1
1070 STA flag
1080 RTS
1090 :
1100 .params   \Window parameters
1110 EQUB 28
1120 EQUB 0 :EQUB 24
1130 EQUB 39:EQUB 0
1140 :
1150 EQUB 28
1160 EQUB 41:EQUB 24
1170 EQUB 79:EQUB 0
1180 ]
1190 NEXT
1200 CALL setup
```

THE ACCOUNT BOOK

Reviewed by Herbert Brothwell

Every business, large or small, needs an effective method of recording its income and expenditure. This can be achieved by purchasing the standard type of account book from any stationery shop.

The drawback of this is that the owner of the business must manually record the details, total each page, review the ledger for bills remaining unpaid, summarise entries for the accounts etc.

A software package, however, is able to do this in a fraction of the time and with greater accuracy. Furthermore, many packages now incorporate management aids to identify bad payers, effect a trial balance, and even show the current level of profitability. The Account Book from Apricote Studios is such a package.

Product	**The Account Book**
Supplier	**Apricote Studios** **2 Purls Bridge Farm,** **Manea, Nr. March,** **Cambs PE15 0ND.** **Tel. (035 478) 432**
Price	**£27.95 inc. VAT**

The program is designed to run on any BBC computer (from the model B to the Master). Although supplied on a 5.25" disc, I actually reviewed the Account Book on a Master Compact by having the program copied onto a 3.5" disc and loading it via the DFS image on the Welcome disc. Assuming that you will be using the 5.25" disc, you will need either: two 80 track single or double sided drives or one 80 track double sided drive.

The package is attractively presented in a plastic folder, and the accompanying manual is well written and makes relatively easy reading. Like all mortals, of course, I gave the manual only a cursory glance before loading the program disc and, surprisingly, found very few problems - a true user-friendly program (I had naturally already made a backup copy of the disc).

This period of 'playing about' with the program is really essential to get the true feel of the Account Book and is strongly recommended. Any problems arising are often solved by reference to the manual, but the very fact that you have spent ten minutes scratching your head makes the point "stick" that much better.

Once I had become reasonably confident in operating the program, I prepared a fictitious set of figures to enter. The main index gives access to four sub-menus, and I entered the Utilities Menu to set up the account name as Landrace Enterprises. Having done that I had to go back to the Main Menu and then again to the Utilities Menu to enter the opening bank and cash balances of the business. The scenario for the fictitious business is given in the table on the next page.

The next stage is probably the most important aspect of the program and needs careful thought. The Receipt and Payment labels are the management aid which will enable you to not only enter receipts and payments quickly, but also to analyse the figures. You can, for instance,

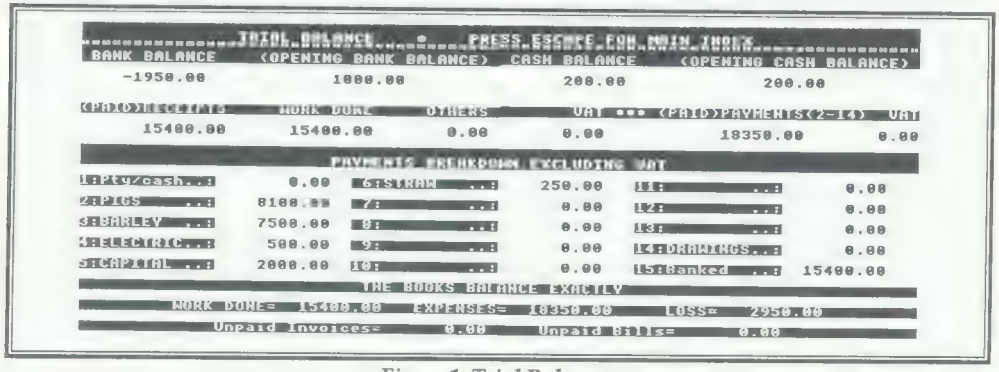


Figure 1. Trial Balance

get a print-out of all the receipts from a particular customer, or details of all receipts in a particular month, or even which bills remain unpaid. For the Receipt label I entered only A=abattoir (the purchasers of pigs) and M=mill (purchaser of the 4 tons of barley).

Landrace Enterprises

Opening bank balance £1000 Opening cash balance £200

		Cost	Receipts
30.04.88	purchases 200 pigs	£4000	
30.04.88	purchases 35 tons of barley	£3500	
30.05.88	pays electricity bill	£ 200	
30.05.88	buys a tractor	£2000	
30.05.88	buys straw	£ 100	
30.10.88	sells 200 pigs		£10000
30.10.88	sells 4 tons barley		£ 400
01.11.88	purchases 200 pigs	£4100	
01.11.88	purchases 30 tons of barley	£4000	
30.12.88	pays electricity bill	£ 300	
30.12.88	buys straw	£ 150	
04.04.89	sells 150 pigs (50 pigs died!)		£ 5000
		Bankrupt	

The Payment labels are broken into two different types. At the top of the screen are the headings for standing orders and frequently recurring payments. I entered A=Pedigree Pigs (source of the pigs) and C=Corn Sellers (source of the barley). The bottom of the screen consists of 15 boxes to enable you to break down your payments into different categories. Boxes 1 and 15 are marked Petty Cash and Banked respectively, and these cannot be changed (the manual does explain why this is necessary). Boxes 2 to 14 however give you plenty of scope to itemise your payments.

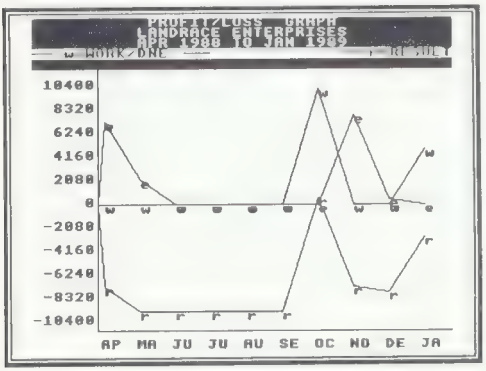


Figure 2. Profit/Loss Graph

This is the only time when I would suggest that you contact your accountant (if you have one), as he would be able to advise you on the categories to use (possibly based on your previous accounts). For Landrace Enterprises I have entered:

2 = pigs
3 = barley
4 = electricity
5 = capital (assets of the business, i.e. tractor)
6 = straw
14 = drawings

Having finally set the system up, I entered all the transactions and then went to the Results Menu. The Trial Balance (Fig.1) reveals (surprise) that Landrace Enterprises made a loss of £2950. This would not be the actual tax loss shown when the final accounts are prepared because, at the end of the day, the firm still has the tractor.

Finally, on the Profit/Loss Graph (Fig.2) we can see that the 'result' line of the graph shows that Landrace Enterprises only came into profit once, in October 1988.

CONCLUSIONS

The Account Book is designed for small businesses which use one bank account. Whilst larger organisations may require more from a software package (e.g. stock control), the small business or self-employed person could find the program invaluable.

The program is completely user-friendly and requires no accounting knowledge. The system is based on the double-entry method of book-keeping and ensures that no error can be made which is not immediately apparent from viewing the Trial Balance.

The various statements which can be printed out will ensure that preparation of your annual accounts will be easier, and if you have an accountant, cheaper than before (bearing in mind that accountants charge around £40 per hour, the package will pay for itself in the first year).

As well as the features already mentioned the Account Book offers:

***full AutoVat facilities - figures for quarterly Returns prepared.**
***search by comment facility.**
***full use of the red function keys .**
***free helpline service.**

The introduction to the manual states "the purpose of this program is to replace the standard type of account book used by most small businesses and the self-employed with a computer version" - it succeeds admirably. B

By Mike Williams and David Spencer

This month we shall be describing and explaining the remaining procedures needed to form our embryo database program. However, before launching into new territory, let's just recap on what we are trying to achieve and where we have got to.

The first two articles in this series developed a simple file handling program to provide the basic requirements of adding, deleting, amending and displaying the records within a file. That program worked well, but suffered from a distinct lack of flexibility. For example, only the particular file specified explicitly in the program could be handled.

To overcome this lack of flexibility we have been developing a set of procedures which could form the basis of a much more general approach. The resulting program could then be used to create and maintain many different files each storing quite different data. The key to this approach is the use of a File Description Record (abbreviated to FDR).

This is a special record at the head of every file created with this approach, which contains a description of the file structure (information such as the size of the file, the number of records stored, and the names, and lengths of all the data fields).

The other key feature of our approach to file handling is the use of an array (called Record$) to provide an internal record area or buffer. Thus entering a new record involves keyboard input into the buffer, followed by the writing of the buffer contents to the file. Displaying a record involves reading a record from the file into the buffer, and then displaying the buffer contents on the screen.

Of course, there is considerably more involved in practice in putting these ideas into effect, and you may well find it useful to re-read the last two articles before continuing further. The procedures we have written so far rely on a set of global variables and arrays (global meaning that they can be accessed directly from within any procedure or function), and those we have introduced so far are listed in table 1.

GLOBAL VARIABLES

Fname$()	field names
Fwidth%()	field widths
Ftype%()	field types
Record$()	record buffers
FH%	number of 256-byte blocks in the File Description Record
FS%	maximum file size in records
NR%	current number of records
RS%	record size in bytes
NF%	number of fields
F%	file channel number
start%	start address of first record

Table 1. List of global variables.

So far we have described procedures to handle the following functions:

Create a file
Open a file
Read a record
Write a record
Enter a record
Display a record
Close a file

These procedures formed the basis of a complete working demonstration program which was included on last month's magazine disc.

To complete a working set of procedures we need two further functions, just as we did with the very first program in this series: how to amend records and how to delete records. Let's start by dealing with record deletion.

DELETING A RECORD

This immediately raises a number of problems, and we will have to decide how to deal with these. The reason for this state of affairs is that it is seldom practical to physically delete a record on demand. If a record is no longer required, what should we do about the physical space it occupies in the file?

We could, of course, blank out all the fields, but the record would still physically exist, and as such could be displayed just like any other. We could mark or flag the record in some way and write our procedures to check for and omit any record so marked. That would still leave a gap in the sequence of valid record numbers.

You might feel that the solution is to remove the empty record by closing up the gap, but it would be very time consuming to do this after every record deletion. Of course, if each record were linked to the next by some form of pointer, then a record could very easily be removed by altering the pointers to bypass the deleted record. But that cannot be applied here.

In fact there is no easy or simple solution. To some extent we have brought the problem upon ourselves by our overall approach of using a serial file, with a record's relative position within the file providing the means of access. As already indicated, some form of linked record structure would make the solution of our current problem much easier, but only at the expense of even greater complexity in file handling as a whole.

We have to make some decisions. First of all we will mark a deleted record in some way and amend our procedure for displaying records to ignore 'deleted' records. Second, we will extend our previous procedure for closing a file so that at that time records are shuffled up to remove any deleted records. In effect, a record will first be logically deleted, and then on closing the file all logically deleted records will be physically deleted. At that stage too, the information on the number of records in the file will be updated. Attempting to change this sooner would cause problems when trying to access any records after those deleted.

To mark a record as logically deleted we will set the top bit of the first byte of the record. Normally, characters have ASCII codes in the range 0 to 127 (see your User Guide). Setting the top bit is equivalent to adding 128 to the ASCII value. So we will have to read the record, set the top bit of the first byte (character), and then write the record back to the file. We will then need to amend our procedure for displaying records to check the first byte of any record, and display only those records for which this byte's value is less than 128.

In a working program, the sequence to be followed would be to ask for the number of the record to be deleted, read this into the buffer and display it on the screen, and then seek confirmation that this is the correct record to be deleted. If confirmed, the record would be marked as a 'deleted' record. Reading and displaying the record would be accomplished easily with existing procedures.

These steps can be written in a 'pseudo-programming' language (quite a useful idea this), and then rewritten in Basic when they are thought to be correct. Pseudo-code is just abbreviated English which you make up to suit your needs. This is illustrated below.

```
=>Check file open - if not display message
      and exit.
=>Get number of record to delete
=>Check valid record number - if not
      display message and exit.
=>Read record
=>Display record
=>Confirm deletion - if yes, logically
      delete record.
=>Exit
```

The revised PROCdisplay_record and the new PROCdelete_record are listed below.

```
 3500 DEF PROCdisplay_record(R%,buffer%)
 3510 CLS:PROCread_record(R%,buffer%)
 3520 IF ASC(Record$(1,buffer%))<128 THEN
PROCdisplay_fields(R%,buffer%) ELSE PROC
error(4)
 3524 ENDPROC
 3526:
 3528 DEF PROCdisplay_fields(R%,buffer%)
 3530 LOCAL I%
 3532 FOR I%=1 TO NF%
 3540 PRINT Fname$(I%);TAB(12);FNfield(R
ecord$(I%,buffer%),Ftype%(I%),pad$)
 3550 NEXT
 3560 ENDPROC
 3570:
 4500 DEF PROCdelete_record(R%,buffer%)
 4510 LOCAL r$
 4520 PROCread_record(R%,buffer%)
 4530 r$=Record$(1,buffer%)
 4540 Record$(1,buffer%)=CHR$(ASC(r$)OR1
28)+MID$(r$,2)
 4550 PROCwrite_record(R%,buffer%)
 4560 delete%=delete%+1
 4570 ENDPROC
```

The display procedure has been modified as described to check a record before displaying it on the screen. For convenience, this process has now been divided into two separate procedures. The second procedure is the one which displays the fields on the screen. If the record has been marked for deletion, then an error routine (PROCerror) is called to display a suitable message ("No such record" for example). All these features, with some slight variations, have been incorporated in the working demo on the magazine disc.

Using an error procedure means that a set of error messages can be easily built up avoiding duplication, and any message can be displayed by calling the routine with an appropriate number.

One other point to note is the introduction of one further global variable, delete%, into PROCdelete_record. This would initially be set to zero. Each time, while a file is open, that a record is deleted, this variable would be incremented by 1. We shall see how this is used in the revised version of PROCclose_file in a moment. At its simplest it acts as a flag which tells us whether any record has been deleted or not.

FILE RE-ORGANISATION

The counterpart to the logical deletion of records described above is the reorganisation of the file when it is closed to physically remove the records marked for deletion. The process is quite straightforward and can be expressed again as pseudo-code before writing it in Basic.

```
=>Check 'delete' flag to see if any
      records have been deleted - if not
      exit.
=>Initialise 'read' and 'write' pointers.
=>Repeat
=>Read next record.
=>If not 'deleted' write record to file,
      and advance 'write' pointer else
      decrement 'delete' flag.
=>Advance 'read' pointer.
=>Until 'delete' flag zero.
=>Copy any remaining records.
=>Exit.
```

The previous FNclose_file has now been re-written as a procedure.

```
 4000 DEF PROCclose_file
 4010LOCAL nr%:nr%=NR%-delete%
 4020 IF delete%=0 THEN PROCexit:ENDPROC
 4030 pread%=1:pwrite%=1
 4040 REPEAT
 4050 PROCread_record(pread%,0)
 4060 IF ASC(Record$(1,0))<128 THEN PROC
write_record(pwrite%,0):pwrite%=pwrite%+
1 ELSE delete%=delete%-1
 4070 pread%=pread%+1
 4080 UNTIL delete%=0
 4090 IF pread%>NR% THEN NR%=nr%:PROCexi
t:ENDPROC
 4100 REPEAT
 4110 PROCread_record(pread%,0)
 4120 PROCwrite_record(pwrite%,0)
 4130 pread%=pread%+1:pwrite%=pwrite%+1
 4140 UNTIL pread%>NR%
 4150 NR%=nr%:PROCexit
 4160 ENDPROC
 4170:
 4180 DEF PROCexit
 4190 PTR#F%=0
 4200 PRINT#F%,FH%,FS%,NR%
 4210 ENDPROC
```

You should be able to follow the programming by matching it against the pseudo-code. Most of the previous function now reappears as a short subsidiary procedure for convenience.

These additional procedures have been incorporated into a new version of the demo program from last month's magazine disc/tape. Please remember that it is only a demonstration. In any such program that is going to be used for real, you will find it necessary to incorporate more checks to avoid the possibility of errors arising (asking for a non-existent record, for example).

UPDATING RECORDS

The other task we have set ourselves is to allow for records to be updated. In principle this is also fairly simple, and again it pays to write it in pseudo-code first.

```
=>Get number of record to update.
=>Read record.
=>Display Record.
=>Update record on screen and in memory.
=>If confirmed write record to file.
```

The hardest part of this is the updating of the screen display and the corresponding buffer contents, and keeping these the same. The screen part of this will depend on what mode you choose to use, and on your design of screen layout. You will need your own input function to deal with this character by character, one field at a time.

You could of course make life simple for yourself by displaying the original record on the screen, and then using our procedure PROCenter_record to input a complete replacement. However, it is much more satisfying to be able to update the the old record directly on the screen. How sophisticated you make this is up to you. We have chosen a fairly simple approach, and a suitable procedure would look something like that listed at the head of the next column.

Another alternative, used by some commercial software, is to reserve part of the screen for data

```
5000 DEF PROCupdate_record(R%,buffer%)
5010 LOCAL I%
5020 PROCdisplay_record(R%,buffer%)
5030 FOR I%=1 TO NF%
5040 PRINTTAB(12,I%-1);
5050 Record$(I%,buffer%)=FNinput(Ftype%
(I%),Fwi dth%(I%),pad$)
5060 NEXT:PRINT
5070 IF FNconfirm THEN PROCwrite_record
(R%,buffer%)
5080 ENDPROC
```

entry purposes. This could be two or three lines at the foot of the screen, or a window, depending on the system. User input to each field appears first in the data entry area until confirmed, whereupon it is transferred to the main record display. This can be used, of course, for both initial data entry and updating.

In fact, if you look at all the procedures we have written you will begin to find quite an overlap between some of them. For example, the only difference between initial data entry and updating is that the existing contents of a record are displayed on the screen first when a record is to be updated. Thus we could write a single procedure to cover both requirements, with an extra parameter in the form of a flag which can be used to determine whether an existing record is to be displayed or not.

Looking for duplication and redundancy in any programs which you write will enable you to develop programs which are both more elegant in their style and which save memory space into the bargain.

We have now reached another turning point in our discussion of file handling. The procedures and functions from this and the last two articles form the basis of a reasonable database program, and a demo program including all our routines is on this month's magazine disc.

Next time we will be taking a look at some new Basic functions related to file handling, and see how these can help improve our programs further. B

BEEBUG WORKSHOP Linked Lists-Part 3

David Spencer concludes this short series on linked lists with a complete heap management system, and gives details of how Basic uses this powerful system.

HEAPS OF MEMORY

In the last two workshops, whenever memory was needed to create a linked list, it was claimed using DIM. Once any part of that memory had been used, it could not be re-used for another purpose. In a real system this method is not practical, and instead something called a 'Heap' is used.

A heap is merely an area of memory from which blocks of differing sizes can be claimed. Once a block has been claimed it can be used for any purpose, for example, storing a linked list. When a particular block of memory is no longer needed it can be returned to the heap and re-allocated later.

The Basic interpreter uses just such a heap to store all the variables used by a program. We will look at this in more detail later, but first we shall show how a heap can be created and manipulated.

HEAP MANAGER

The software responsible for the creation of a heap, and the subsequent allocation and de-allocation of blocks of memory, is called a 'Heap Manager'. It is such a heap manager that we will develop now. If you ignore the de-allocation of memory, the heap manager is really very simple. You maintain a pointer to the first free byte of memory, and whenever a block is requested, return the value of this pointer, and increment it by the size of the new block. If there is not sufficient memory to allocate to the new block, the heap manager returns an error instead.

The process of handling blocks of memory that have been returned to the heap manager when they are no longer needed is more difficult. When a block is de-allocated it will leave a gap in the middle of the heap, unless it happens to have been the last block to be allocated. It would be possible to close up this hole by moving the remainder of the heap downwards. The problem with this is that the data in the blocks that are moved might rely on being at a certain address, and also, moving large amounts of memory can be time consuming. The way round this problem is to make all the free blocks into a linked list themselves. Each free block then contains details of its length, and a pointer to the next free block. The final block will contain a null pointer.

Using this method, whenever the heap manager is required to allocate a new block, it simply needs to go through the list and find a suitable sized block. Once this has been done, the linked list of free space can be re-built.

HEAP DESCRIPTOR

Our heap in memory needs to contain certain items of information. The most important item to store is the pointer to the list of free blocks. It would also be useful to identify the memory being used as being part of a heap area. All this information is stored in a heap descriptor at the start of the heap memory. Our heap manager will use a heap descriptor consisting of two four-byte entries. The first will be the characters 'HEAP' to identify the area as a heap, and the second will be the list head. This will contain a pointer to the first free block, or null if no blocks are available.

BLOCK FORMAT

Each block of memory allocated by our heap manager will contain, in the first four bytes, the

length of that block. The length allocated is four bytes longer than that requested, in order to allow for this. The address of the block that is returned to the application is in fact the address of the first byte after the length.

USING THE HEAP MANAGER

listing 1 contains the three routines that make up our heap manager, together with the word sort program from last month's Workshop. Provided you have followed our look at linked lists, you should be able to see how this program works by comparing it with the one from last month.

It is very easy to use the heap manager functions in your own programs. There are three functions:

```
heap=FNheap_create(size)
```

will create a new heap of length 'size' (in bytes) and return its address. To claim a block of memory use:

```
block=FNheap_claim(heap,size)
```

where 'heap' is the address of the heap, and 'size' is the length required; and to release a block use:

```
valid=FNheap_release(heap,block)
```

where 'heap' is the address of the heap, and 'block' is the address of the block to be freed. All these calls return zero if the operation could

Listing1

```
  10 REM Program Word Sort with Heap
  20 REM Version B 1.0
  30 REM Author  David Spencer
  40 REM BEEBUG  October 1988
  50 REM Program subject to copyright
  60 :
 100 heap=FNheap_create(1000)
 110 head=FNheap_claim(heap,4):!head=0
 120 PRINT"ENTER WORDS"
 130 REPEAT:INPUT word$
 140 IF word$<>"" PROCenterword(head,wo
rd$)
 150 UNTIL word$=""
 160 PRINT'"DELETE WORDS"
 170 REPEAT:INPUT word$
 180 IF word$<>"" THEN IF NOT FNremovew
ord(head,word$) PRINT "Word not in list"
:VDU 7
 190 UNTIL word$=""
 200 ptr=head
 210 REPEAT
 220 ptr=!ptr
 230 IF ptr THEN PRINT$(ptr+8);TAB(20);
ptr!4
 240 UNTIL ptr=0
 250 END
 260 :
1000 DEF PROCenterword(ptr,A$)
1010 LOCAL new
1020 IF !ptr=0 THEN 1050
1030 IF $(!ptr+8)=A$ THEN !(!ptr+4)=!(!
ptr+4)+1:ENDPROC
1040 IF $(!ptr+8)<A$ THEN PROCenterword
(!ptr,word$):ENDPROC
1050 new=FNheap_claim(heap,LEN(A$)+9)
1060 !new=!ptr:new!4=1:$(new+8)=A$
1070 !ptr=new
1080 ENDPROC
1090 :
1100 DEF FNremoveword(ptr,A$)
1110 LOCAL ptr2:IF !ptr=0 THEN =FALSE
1120 IF $(!ptr+8)=A$ THEN ptr2=!ptr:!pt
r=!ptr2:=FNheap_release(heap,ptr2)
1130 =FNremoveword(!ptr,A$)
1140 :
10000 DEF FNheap_create(len)
10010 LOCAL heap
10020 IF len<16 THEN =0
10030 DIM heap len-1:$heap="HEAP"
10040 heap!4=heap+8:heap!8=0
10050 heap!12=len-8
10060 =heap
10070 :
10080 DEF FNheap_release(heap,add)
10090 LOCAL ptr:add=add-4
10100 IF !heap<>&50414548 THEN =FALSE
10110 add!4=!add:!add=0
10120 ptr=heap+4
10130 REPEAT IF !ptr THEN ptr=!ptr
10140 UNTIL!ptr=0
10150 !ptr=add:=TRUE
10160 :
10170 DEF FNheap_claim(heap,len)
10180 LOCAL ptr,ptr2,bptr,max,size
10190 IF !heap<>&50414548 THEN =FALSE
10200 len=len+4:ptr=heap+4:max=&FFFF:IF
len<8 len=8
10210 REPEAT
10220 ptr2=!ptr:size=ptr2!4
10230 IF ptr2<>0 AND size>=len AND size<
max THEN max=size:bptr=ptr
10240 ptr=ptr2
10250 UNTIL ptr=0
10260 IF max=&FFFF THEN =0
10270 ptr=!bptr:ptr2=!ptr:!ptr=len
10280 IFmax-len<8 THEN !bptr=ptr2:=ptr+4
10290 ptr!len=ptr2:ptr!(len+4)=max-len
10300 !bptr=ptr+len:=ptr+4
```

not be performed for some reason, such as insufficient free memory to allocate a block.

BASIC AND LINKED LISTS

So far, we have explained the theory of linked lists and given some rather abstract examples of their use. So as to not leave you thinking "Yes, all very nice but do they have a real use?", we will end our look at linked lists by explaining how Basic uses a heap to store its variables.

One of the easiest ways for Basic to store variables would be to place the name of each variable, followed by its value, in memory, one after another. However, searching for and altering any variable would be very time consuming using this method.

You should already be thinking by now that a better solution to the variable storage problem would be to use a linked list. By storing a pointer with each variable name and value as well, you could join together all variables to form a linked list. Now, searching for a variable would be much quicker, because you can find the next variable without searching through all the data that makes up the current variable. This also means that the variables do not need to be stored one after another.

However, even this technique is not fast enough for BBC Basic. The problem is that the linked list of variables can still get very long, and it will take a relatively long time to find a variable at the end of the list. The solution to this is to have several short lists, rather than one long one. Basic uses the first character of any variable name to choose one of fifty four lists. Each of these lists has a list head which is simply a two byte pointer containing the address of the first variable in that list. If this address is zero (or in practice any value less than 256) then that list is empty. These pointers are stored in memory at locations &400+2*x (low byte) and &401+2*x (high byte), where 'x' is the ASCII code of the first letter. For example, the list for variables beginning with the letter 'A' is at locations &482 and &483.

Within each list, each variable is stored with the first two bytes being a pointer to the next variable in that list, or zero if at the end of the list. Then comes the variable name, without its first letter, then a zero byte, and finally the variable's value. The format of this depends on the type of variable. With this method, assuming that the spread of 'first letters' is uniform, each list will be only 1/54th the size of a single list. Locating a variable is much quicker as a result.

The program in Listing 2 prints out the names of all Basic variables defined. By adding more assignments in PROCdefine, you will see more names printed out. It should be easy to follow the working of this program. Once Basic has created a variable, its value is not discarded until *all* the variables are cleared. Regrettably, Basic does not implement a true heap management system, and memory allocated for local storage or for string storage is not recovered when discarded by a program. String storage in particular can waste large areas of memory if allowed to by the programmer.

Listing2

```
   10 REM Program Variable List
   20 REM Version B1.0
   30 REM Author  David Spencer
   40 REM BEEBUG  October 1988
   50 REM Program Subject To Copyright
   60 :
  100 PROCdefine
  110 FOR F%=ASC"A" TO ASC"z"
  120 P%=!(&400+F%*2) AND &FFFF
  130 PROClist(F%,P%)
  140 NEXT
  150 END
  160 :
 1000 DEF PROClist(first,ptr)
 1010 IF ptr<256 ENDPROC
 1020 VDU first:O%=1
 1030 REPEAT O%=O%+1
 1040 IF ptr?O%<>0 VDU ptr?O%
 1050 UNTIL ptr?O%=0:PRINT
 1060 PROClist(first,!ptr AND &FFFF)
 1070 ENDPROC
 1080 :
 1090 DEF PROCdefine
 1100 REM Put sample definitions here
```

That ends our look at linked lists, but next month we will continue on the theme of data structures by looking at trees. B

Z88 MODEM REVIEW

David Spencer puts his Z88 on-line and finally becomes a fully-fledged yuppie.

Product	Z88 Pocket Modem
Supplier	Cambridge Computer, Bridge House, 10 Bridge Street, Cambridge CB2 1UE. Tel. (0223) 312216
Price	£179.95 inc. VAT and P&P

The first thing to say about the new Z88 modem is that it is not the elegant black box carrying Cambridge Computer's name as originally promised. Instead, it is a standard Datatronics 1200P, dual speed auto-dial auto-answer modem, in a beige case. The overall size of the device is about 3" by 2" by 1" deep, with a 25 way D-connector protruding from one end. The back of the case carries two American style phone connectors, one for the phone line, the other for the phone, and also a 2.5mm socket for a 9V power supply, this latter overriding the internal 9V battery. On the top of the box are three red lights, one for low battery, one to indicate the baud rate, and one to show the presence of a carrier. The modem does not yet have BABT approval. This does not mean that the modem is in anyway substandard or dangerous, rather that the device has not yet completed the strict approval system.

Supplied with the modem are two leads, one to connect it to the Z88, the other to connect to the phone line, the driving software in a 32K EPROM card, and manuals for both this and the modem.

The technical specification of the 1200P modem is quite impressive. The only real problem is that while 300/300 and 1200/1200 baud are supported, 1200/75 is not. This doesn't cause any problems with Prestel, which now uses multi-speed Vasscom ports, but it may prevent the modem being used with other viewdata systems. The 1200P is fully Hayes compatible, which means that it is controlled by a standard set of commands, and will therefore work with a wide range of communications software. Features such as tone dialling and last number re-dial are also supported.

The software supplied with the modem is a cut down version of Wordmonger's Zterm package, renamed Comm88. Following the fashion of other Z88 software, all the functions of the modem are controlled through menus or 'diamond key' sequences and help information is provided for all the commands. There are three modes of operation. Two of these are viewdata and plain text (called teletype), and the third allows commands at the lowest level to be sent directly to the modem. Comm88 can be configured for such details as the mode to use, the baud rate and the data format, and this configuration is automatically saved to a file. Additionally, up to five function keys can be programmed with passwords, log-on sequences etc.

Comm88 offers the option to dial a number from a stored list, or entered from the keyboard. When using a stored list, the correct baud rate and mode will be selected automatically. The viewdata mode is fairly primitive, with each page being split into three sections to fit on the display. Graphics are only partly supported, and obviously there is no colour. There is also no facility to download telesoftware. The teletype mode does not suffer from these problems, and includes two-way file transfer using the XMODEM standard.

In conclusion, the modem performed very well in use, but is clearly somewhat limited by the Z88 - I would not want to read many Prestel pages in sections on the eight line display. The price of nearly £180 pounds is also rather high, acceptable, maybe, to the jet-setting yuppie executive, but rather excessive for ordinary mortals like you and I. B

FIRST COURSE (Continued from page 32)

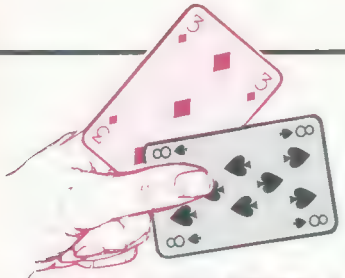
```
 1600 PRINTTAB(26-x%,pos%);name$;SPC8;TA
B(26,pos%);name$
 1610 PROCwait(PA)
 1620 NEXT
 1630 ENDPROC
 1640 :
 1650 DEF PROCuandd(from%,to%,stp%,at%,v
al%)
 1660 name$=LEFT$(name$(from%)+STRING$(7
," "),7)
 1670 FOR y%=from% TO to% STEP stp%
 1680 PRINTTAB(at%,y%+val%);SPC8
 1690 PRINTTAB(at%,y%);name$
 1700 PROCwait(PA)
 1710 NEXT
 1720 ENDPROC
 1730 :
 1740 DEF PROCright(pos%,to%,at%,aorc%)
 1750 name$=LEFT$(name$(pos%)+STRING$(7,
" "),7)
 1760 FOR x%=1 TO to%
 1770 PRINTTAB(at%+x%,aorc%);" ";name$
 1780 PROCwait(PA)
 1790 NEXT
 1800 ENDPROC
```

```
 1810 :
 1820 DEF PROCcullen
 1830 REPEAT:finish%=TRUE
 1840 FOR b%=1 TO max%-1
 1850 FOR s%=b%+1 TO max%
 1860 IF name$(b%)>name$(s%) THEN PROCsw
ap(b%,s%):finish%=FALSE
 1870 NEXT:NEXT
 1880 UNTIL finish%
 1890 ENDPROC
 1900 :
 1910 DEF PROCselect
 1920 FOR b%=1 TO max%-1
 1930 b$=name$(b%):found%=FALSE
 1940 FOR s%=b% TO max%
 1950 IF b$>name$(s%) THEN found%=TRUE:b
$=name$(s%):d%=s%
 1960 NEXT
 1970 IF found% PROCswap(b%,d%)
 1980 NEXT
 1990 ENDPROC
 2000 :
 2010 DEF PROCwait(t)
 2020 t=INKEY(t)
 2030 ENDPROC
```

THE BEEBUG SUPER-SQUEEZE (Continued from page 49)

```
 3710 LDA newv+2:BEQ fnnn5:JSR badd
 3720 .fnnn5 JMP chew45
 3730 .chew19 CMP #&8B:BNE noel:PHA
 3740 LDA #0:STA eflg:PLA:.noel CMP #&E7
 3750 BNE noif:DEC tflg:.noif CMP #&85
 3760 BNE noerrr:DEC oeflg
 3770 .noerrr CMP #&F4:BEQ chewdn
 3780 JSR tokchk:BNE noexp
 3790 LDA #&FF:STA eflg
 3800 .noexp LDA (lptr),Y:JSR badd
 3810 JMP chew4:.chewdn:LDA #13:JSR badd
 3820 LDA boff:STA sbuff+2:JSR nxtlin
 3830 SEC:LDA sbuff+2:SBC #3:CLC
 3840 ADC lbuff+2:BCS tolo
 3850 .notol LDA sbuff+3:CMP #&DC
 3860 BEQ tolo:CMP #&DD:BEQ tolo
 3870 LDA lnos:STA temp:LDA lnos+1
 3880 STA temp+1:.clin LDY #0:JSR clinc
 3890 LDA temp+1:CMP #&C0:BEQ app
 3900 LDA (temp),Y:CMP sbuff+1:BNE clin2
 3910 INY:LDA (temp),Y:CMP sbuff
 3920 BEQ tolo:.clin2 JSR clinc:JMP clin
 3930 .tolo:JSR shunt:LDA #&80:STA lbuff
 3940 .app LDA lbuff:CMP #&80:BEQ app2
 3950 LDX lbuff+2:LDA lbuff-2,X
```

```
 3960 CMP #ASC":":BNE app1:DEX
 3970 .app1 LDA #ASC":"
 3980 STA lbuff-1,X:LDY #3
 3990 .app3 LDA sbuff,Y:STA lbuff,X
 4000 INY:INX:CMP #13:BNE app3
 4010 STX lbuff+2:.app5 BIT tflg
 4020 BMI pback:BIT sflg:BMI pback
 4030 BIT oeflg:BMI pback:LDA sbuff+3
 4040 CMP #&DC:BEQ pback:JMP chew
 4050 .app2 LDY #0:.app4 LDA sbuff,Y
 4060 STA lbuff,Y:INY:CPY sbuff+2
 4070 BNE app4:STY lbuff+2:JMP app5
 4080 .pback JSR shunt:LDA #&80
 4090 STA lbuff:JMP chew
 4100 .clinc INC temp:BNE clinc2
 4110 INC temp+1:.clinc2 RTS
 4120 .sbuff EQUS STRING$(128,CHR$0)
 4130 EQUS STRING$(128,CHR$0)
 4140 .vbuff EQUS STRING$(128,CHR$0)
 4150 EQUS STRING$(128,CHR$0)
 4160 .lbuff EQUS STRING$(128,CHR$0)
 4170 EQUS STRING$(128,CHR$0)
 4180 .end:]NEXT
 4190 OSCLI ("SAVE SQZOBJ "+STR$~code+"
"+STR$~O%+" 8000 8000")
```



'ELEVENSES'

Paul Timson presents an excellent implementation of the popular card game Elevens, a variation of patience.

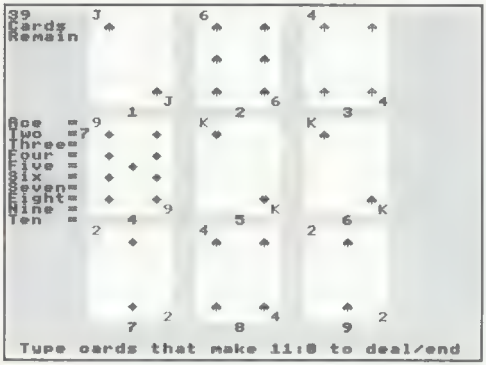
For those that are not already familiar with the game of Elevens, the idea is to lay down the entire pack of cards on the table by obeying certain rules. Firstly, nine cards are laid face down on the table, in a three by three grid. Each card is turned over until two cards are turned that add up to eleven. These two cards are then covered with two new cards from the pack placed face upwards. Should these two cards then add up to eleven they must be covered again in the same manner. If a Jack, Queen, and King appear at the same time they may also be covered with three new cards from the pack. The game ends when there are no more cards to turn and no two cards add up to eleven. You win by using up all the cards in the pack, by no means an easy feat.

Type in the listing as printed and save it before going any further. Note that PAGE must be reduced to &1200 before running the program if you do not have shadow memory. When the program is run it will first inform you that the pack is being shuffled. The nine cards will then be dealt, face down, on the table. Each card will be identified with a number between one and nine. Press the 0 key to turn the first card, and again to turn the second. If these two cards add up to eleven then press the keys corresponding to their identifying numbers (i.e. 1 and 2 in this case). If this is not the case then continue turning cards with the 0 key until two cards add up to eleven. Should a Jack, Queen, and King appear in any order, these may also be covered in the same manner. The game will end when there are no more possible moves or when you have used all of the cards in the pack.

Although the game is extremely easy to play and very simple, it is visually entertaining and quite addictive. Type it in and give it a go.

```
   10 REM Program Elevens
   20 REM Version B1.03
   30 REM Author  Paul Timson
   40 REM BEEBUG  October 1988
   50 REM Program Subject To Copyright
   60 :
  100 ON ERROR MODE 7:REPORT:PRINT" at l
ine ";ERL:END
  110 MODE1: PROCdefine: PROCinit
  120 REPEAT:CLS
  130 PROCset
  140 REPEAT
  150 COLOUR0:COLOUR130
  160 FORline%=1TO10:PRINTTAB(0,9+line%)
;SPC(7):NEXTline%
  170 PRINTTAB(0,0);deck%;SPC(1);TAB(0,1
);"Cards";TAB(0,2);"Remain":first%=GET-4
8
  180 IF first%>=1 AND first%<=cards% AN
D cards%>=2 PROCcards
  190 IF first%=0 PROCdeal:IF cards%=10
deck%=deck%+1:third%=-1:PROCfinished
  200 IF deck%=1 AND cards%=9 third%=-1:
PROCfinished
  210 IF deck%=0 PROCend
  220 UNTILdeck$="!"
  230 PRINTTAB(5,24);"Do you want anothe
r go (Y/N)?"
  240 key$=GET$:IFkey$<>"y"ANDkey$<>"n"A
NDkey$<>"Y"ANDkey$<>"N" GOTO240
  250 UNTIL NOT(key$="y" OR key$="Y")
  260 MODE7
  270 END
  280 :
 1000 DEF PROCshuffle
 1010 deck$="":FOR shuffle%= 51 TO 0 STE
P-1
 1020 random%=RND(shuffle%):deck$=deck$+
MID$(shuffled$,2*random%+1,2):shuffled$=
LEFT$(shuffled$,random%*2)+RIGHT$(shuffl
ed$,(shuffle%-random%)*2)
 1030 NEXTshuffle%:shuffled$=deck$
 1040 ENDPROC
 1050 :
 1060 DEF PROCinit
 1070 DIMdealt$(9)
 1080 VDU23,1,0;0;0;0;19,2,2,0,0,0
 1090 *FX11,0
 1100 shuffled$="AHACADAS2H2C2D2S3H3C3D3
S4H4C4D4S5H5C5D5S6H6C6D6S7H7C7D7S8H8C8D8
S9H9C9D9STHTCTDTSJHJCJDJSQHQCQDQSKHKCKDK
S"
 1110 PROCshuffle:PROCshuffle:ENDPROC
 1120 :
 1130 DEF PROCset
 1140 COLOUR130:CLS:COLOUR0:PRINTTAB(14,
5);"E L E V E N S";TAB(13,15);"By  Paul
```

```
Timson";TAB(15,25);"SHUFFLING!!"
 1150 PROCshuffle:PROCshuffle
 1160 PRINTTAB(14,5);SPC(13);TAB(13,15);
SPC(15);TAB(15,25);SPC(11);TAB(10,9);"1"
;TAB(19,9);"2";TAB(28,9);"3";TAB(10,19);
"4";TAB(19,19);"5";TAB(28,19);"6";TAB(10
,29);"7";TAB(19,29);"8";TAB(28,29);"9"
 1170 PRINTTAB(1,31);"Type cards that ma
ke 11:0 to deal/end";
 1180 temp$=deck$:deck$="**************
***":cards%=0:REPEAT:PROCdeal:UNTILcards
%=9:deck$=temp$:COLOUR130:COLOUR0:cards%
=0:deck%=52
 1190 ENDPROC
 1200 :
 1210 DEF PROCdeal
 1220 cards%=cards%+1:deck%=deck%-1:IFca
rds%=10THENENDPROC
 1230 IFcards%=10ORcards%=40ORcards%=7 xco
ord%=7ELSEIFcards%=20Rcards%=50Rcards%=8
 xcoord%=16ELSEIFcards%=30Rcards%=60Rcar
ds%=9 xcoord%=25
 1240 IFcards%>=1ANDcards%<=3 ycoord%=0E
LSEIFcards%>=4ANDcards%<=6 ycoord%=10ELS
EIFcards%>=7ANDcards%<=9 ycoord%=20
 1250 card$=LEFT$(deck$,1):deck$=RIGHT$(
deck$,LEN(deck$)-1):suit$=LEFT$(deck$,1)
:deck$=RIGHT$(deck$,LEN(deck$)-1):dealt$
(cards%)=card$+suit$:PROCdisplay
 1260 ENDPROC
 1270 :
 1280 DEF PROCdisplay
 1290 LOCAL temp,temp$,whole$,part,line$
,line,count,char
 1300 IF card$="A" whole$="30*30"ELSEIFc
ard$="2" whole$="9*41*9"ELSEIFcard$="3"
whole$="9*20*20*9"ELSEIFcard$="4" whole$
="7*3*37*3*7"ELSEIFcard$="5" whole$="7*3
*18*18*3*7"ELSEIFcard$="6" whole$="7*3*1
6*3*16*3*7"
 1310 IF card$="7" whole$="7*3*4*11*3*16
*3*7"ELSEIFcard$="8" whole$="7*3*4*11*3*
11*4*3*7"ELSEIFcard$="9" whole$="7*3*9*3
*4*4*3*9*3*7"ELSEIFcard$="T"whole$="6*3*
4*4*3*9*3*4*4*3*6"
 1320 IFcard$="J"ORcard$="Q"ORcard$="K"
whole$="7*45*7"ELSEIFcard$="*" whole$=ST
RING$(63,"*")
 1330 IF card$="*" GOTO1410
 1340 temp$=card$:IFcard$="T" temp$="10"
 1350 whole$=whole$+"*":temp%=0:FORcount
%=1TOLEN(whole$)-1
 1360 IF MID$(whole$,count%,1)<>"*"ANDMI
D$(whole$,count%+1,1)<>"*" temp$=temp$+S
TRING$(VAL(MID$(whole$,count%,1))*10," "
),
 1370 IFMID$(whole$,count%,1)<>"*"ANDMID
$(whole$,count%+1,1)="*" temp$=temp$+STR
ING$(VAL(MID$(whole$,count%,1)),"  ")
 1380 IF MID$(whole$,count%,1)="*" temp$
=temp$+"*"
 1390 NEXTcount%:IFcard$="T" whole$=temp
$+"10"
 1400 IFcard$<>"T" whole$=temp$+card$
 1410 COLOUR131:COLOUR1:IFsuit$="D" char
%=224ELSEIFsuit$="H" char%=225ELSEIFsuit
$="S" char%=226:COLOUR0ELSEIFsuit$="C" c
har%=227:COLOUR0ELSEIFsuit$="*" char%=22
8
 1420 FORline%=0TO8:LETline$=MID$(whole$
,(line%*7)+1,7)
 1430 FORpart=1TO7:IFMID$(line$,part,1)=
"*"THENLETline$=LEFT$(line$,part-1)+CHR$
(char%)+RIGHT$(line$,7-part)
 1440 NEXTpart:PRINTTAB(xcoord%,ycoord%+
line%);line$:NEXTline%
 1450 ENDPROC
 1460 :
 1470 DEF PROCcards
 1480 IFLEFT$(dealt$(first%),1)="J"ORLEF
T$(dealt$(first%),1)="Q"ORLEFT$(dealt$(f
irst%),1)="K"THENIFdeck%>=3 PROCthree:EN
DPROC
 1490 IFLEFT$(dealt$(first%),1)="J"ORLEF
T$(dealt$(first%),1)="Q"ORLEFT$(dealt$(f
irst%),1)="K" ENDPROC
 1500 IFdeck%>=2 PROCtwo
 1510 ENDPROC
 1520 :
 1530 DEF PROCtwo
 1540 sum%=first%:PROCAtoT
 1550 second%=GET-48:IFsecond%=0 ENDPROC
 1560 IFsecond%<10Rsecond%>cards% GOTO15
50
 1570 IFLEFT$(dealt$(second%),1)="*"ORse
cond%=first% GOTO1550
 1580 sum%=0:number%=first%:PROCadd:numb
er%=second%:PROCadd:IFsum%<>11 GOTO1550
 1590 sum%=second%:PROCAtoT:temp=cards%:
cards%=first%-1:PROCdeal:cards%=second%-
1:PROCdeal:cards%=temp
 1600 ENDPROC
 1610 :
 1620 DEF PROCAtoT
 1630 PRINTTAB(0,10);"Ace  =";TAB(0,11);
"Two  =";TAB(0,12);"Three=";TAB(0,13);"F
our =";TAB(0,14);"Five =";TAB(0,15);"Six
  =";TAB(0,16);"Seven=";TAB(0,17);"Eight
=";TAB(0,18);"Nine =";TAB(0,19);"Ten  ="
 1640 IFLEFT$(dealt$(sum%),1)="A" PRINTT
AB(6,10);sum%ELSEIFLEFT$(dealt$(sum%),1)
="T" PRINTTAB(6,19);sum%ELSEPRINTTAB(6,9
+VAL(LEFT$(dealt$(sum%),1)));sum%
 1650 ENDPROC
```




```
 1660 :
 1670 DEF PROCadd
 1680 IFLEFT$(dealt$(number%),1)="A" sum
%=sum%+1ELSEIFLEFT$(dealt$(number%),1)="
T" sum%=sum%+10ELSEsum%=sum%+VAL(LEFT$(d
ealt$(number%),1))
 1690 ENDPROC
 1700 :
 1710 DEF PROCthree
 1720 sum%=first%:PROCJQK
 1730 second%=GET-48:IFsecond%=0 ENDPROC
 1740 IFsecond%<1ORsecond%>cards% GOTO17
30
 1750 IFdealt$(second%)="*"ORsecond%=fir
st%ORLEFT$(dealt$(second%),1)=LEFT$(deal
t$(first%),1) GOTO1730
 1760 IFLEFT$(dealt$(second%),1)<>"J"AND
LEFT$(dealt$(second%),1)<>"Q"ANDLEFT$(de
alt$(second%),1)<>"K" GOTO1730
 1770 sum%=second%:PROCJQK
 1780 third%=GET-48:IFthird%=0 ENDPROC
 1790 IFthird%<1ORthird%>cards% GOTO1780
 1800 IFdealt$(third%)="*"ORthird%=first
%ORthird%=second% GOTO1780
 1810 IFLEFT$(dealt$(third%),1)=LEFT$(de
alt$(first%),1)ORLEFT$(dealt$(third%),1)
=LEFT$(dealt$(second%),1) GOTO1780
 1820 IFLEFT$(dealt$(third%),1)<>"J"ANDL
EFT$(dealt$(third%),1)<>"Q"ANDLEFT$(deal
t$(third%),1)<>"K" GOTO1780
 1830 sum%=third%:PROCJQK:sum%=ASC(LEFT$
(dealt$(first%),1))+ASC(LEFT$(dealt$(sec
ond%),1))+ASC(LEFT$(dealt$(third%),1)):I
Fsum%<>230 ENDPROC
 1840 temp=cards%:cards%=first%-1:PROCde
al:cards%=second%-1:PROCdeal:cards%=thir
d%-1:PROCdeal:LETcards%=temp
 1850 ENDPROC
 1860 :
 1870 DEF PROCJQK
 1880 PRINTTAB(0,10);"Jack =";TAB(0,11);
"Queen=";TAB(0,12);"King =":IFLEFT$(deal
t$(sum%),1)="J" PRINTTAB(6,10);sum%ELSEI
FLEFT$(dealt$(sum%),1)="Q" PRINTTAB(6,11
);sum%ELSEIFLEFT$(dealt$(sum%),1)="K" PR
INTTAB(6,12);sum%
 1890 ENDPROC
 1900 :
 1910 DEF PROCend
 1920 deck$=STRING$(18,"*"):deck%=9:COLO
UR0:COLOUR130:PRINTTAB(1,31);"Now to fin
ish type pairs/JQKs-0 to end";:TIME=0:RE
PEAT:VDU19,2,3,0,0,0:VDU19,2,2,0,0,0:UNT
ILTIME=100
 1930 REPEAT:COLOUR0:COLOUR130:FORline%=
1TO10:PRINTTAB(0,9+line%);SPC(7):NEXTlin
e%:PRINTTAB(0,0);SPC(3)'SPC(5)'SPC(6):LE
Tfirst%=GET-48
 1940 IFfirst%=0 third%=-1:PROCfinished
 1950 IFfirst%>=1 ANDfirst%<=9 ANDdealt$
(first%)<>"*" PROCcards
 1960 third%=0:FOR line%=1TO9:IFdealt$(l
ine%)<>"**" third%=1
 1970 NEXT line%:IF third%=0 COLOUR0:COL
OUR130:FORline%=1 TO10:PRINTTAB(0,9+line
%);SPC(7):NEXTline%:PROCfinished
 1980 UNTILdeck$="!"
 1990 ENDPROC
 2000 :
 2010 DEF PROCfinished
 2020 COLOUR130:COLOUR0:PRINTTAB(0,0);SP
C(3)'SPC(5)'SPC(7);TAB(0,31);STRING$(39,
" ");
 2030 IFthird%=0 PRINTTAB(11,4);"W E L L
   D O N E";TAB(8,14);"You have won at E
levens"
 2040 IF LEFT$(deck$,1)<>"*" AND third%=
-1 PRINTTAB(5,4);"You have ";deck%;" car
d(s) remaining" ELSEIF LEFT$(deck$,1)="*
" ANDthird%=-1 PRINTTAB(12,4);"B A D   L
 U C K";TAB(7,14);"You almost won at Ele
vens"
 2050 IFdeck%=1 AND LEFT$(deck$,1)<>"*"A
NDthird%=-1 PRINTTAB(2,14);"To win you n
eed 1 or 3 JQKs remaining"
 2060 deck$="!"
 2070 ENDPROC
 2080 :
 2090 DEF PROCdefine
 2100 VDU23,224,0,8,28,62,127,62,28,8
 2110 VDU23,225,0,20,62,127,127,62,28,8
 2120 VDU23,226,0,8,28,62,127,127,42,8
 2130 VDU23,227,0,8,28,42,127,42,8,8
 2140 VDU23,228,129,66,36,24,24,36,66,12
9
 2150 ENDPROC
```

In the Z88 Page this month, David Spencer explains the use of windows on the Z88.

The display on the Z88 is divided into three areas as shown in figure 1. Normally, characters can be output to the entire application area. It is, however, possible to define a series of windows on the screen, and then any text will be output to the current window. The operating system allows a total of eight windows, although three of them are reserved for error boxes and pop-up applications. This leaves the user with five windows, numbered 1 to 5. Normally, only window 1 is active, and this is set to cover the whole application area of the screen.

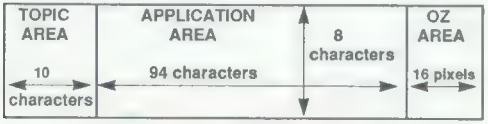


Figure 1. Z88 screen layout.

The definition of a window can be changed using the command:

```
VDU 1,55,35,window,32+x,32+y,32+w,32+d,flags
```

where:

window is the window number (1-5) as an ASCII digit, (window 1=ASC"1"=49 etc.).

x is the x co-ordinate of the top left-hand corner of the window.

y is the y co-ordinate of the top left-hand hand corner.

w is the width of the window in characters.

d is the depth of the window in characters.

flags is a byte that determines how the border of the window appears. There are four possible values:

&80 No border
&81 Vertical bars at side
&82 Shelf brackets at top
&83 Both bars and brackets

The vertical bars provide an edge to the window. Shelf brackets, if selected, form a sort of elbow in both top corners of the window. These allow any reversed text printed on the top line to join onto the side bars. This is how windows are given titles.

The co-ordinates of the top left corner of the window are specified relative to the top left corner of the applications area. However, the 'x' value can be negative, allowing windows to be specified in the topic area of the screen.

The selection of which window is used is performed by one of three different commands. First:

```
VDU 1,50,67,window
```

selects the window (again given as an ASCII digit), clears it, and sets the display toggles (bold etc.) to off. The next command:

```
VDU 1,50,72,window
```

selects the window, but without clearing it, and maintaining the same toggles as when it was last selected. Finally:

```
VDU 1,50,73,window
```

selects a window, and then both clears it and resets the toggles only if it is the first time the window has been selected. Otherwise, the window's contents are left unchanged.

As an example, try the following program:

```
10flags=&83
20VDU 1,55,35,ASC"2",23,32,40,40,flags
30VDU 1,50,67,ASC"2",1,ASC"R"
40PRINT "WINDOW 2"
50VDU 1,ASC"R"
60PRINT "This text is in window number two."
70A=GET:VDU 1,50,72,ASC"1"
```

This program sets up an 8 by 8 window in the topic area, and gives it a reversed title (using VDU 1,ASC"R" to select reverse printing), and some sample text. Pressing any key will re-select the default window number 1. Try altering the value of the variable 'flags' to one of the other options given earlier, and see what effect it has.

Another useful pair of commands is:

```
VDU 1,50,71,43 and
VDU 1,50,71,45
```

The first one of these 'greys' out all of the current window, while the second one 'un-greys' it again. If you grey the current window before selecting another window, the new window will stand out. You must of course un-grey the original window when it is re-selected. This is the method used by pop-ups such as the clock and the calculator. B

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tips

This month's collection of hints and tips are rounded up by Lance Allison. Remember that we are always looking out for good hints. All hints featured on this page are awarded five pounds and we award fifteen pounds for the star hint.

*** * * STAR HINT * * ***

ELAPSED DAYS

by Bernard Hill

The following function will calculate and return the number of days that have elapsed since 30 November 1 BC (Modern Calendar, not pre-Gregorian).

```
1000 DEFFNd(d,m,y)
1010 IF m<3 THEN m=m+12:y
     =y-1
1020 =d+31*m-INT(0.4*m
     +2.3)+365*y+yDIV4
     -INT(0.75+(yDIV100+1))
```

Use `FNd(d1,m1,y1)-FNd(d2,m2,y2)` to calculate the number of days between two dates. Note that d,m, and y are all integer numbers representing the day, month and year, and that the year must be the full year (ie. y=1988 not 88).

To complement that, here is a second function to return the day of the week for a specified date. The day is returned as an integer number in the range of 0 to 6 where 0=Sunday and 6=Saturday. It will work for all dates including leap years and non-century leap years.

```
2000 DEFFNdn(d,m,y) =
       (FNd(d,m,y)+1)MOD 7
```

For example, FNdn(19,8,1988) will return 5 for Friday.

VARIABLE CURSOR

by Brian Care

The following VDU sequence will generate a flashing cursor whose height is dependent on the value n.

```
VDU 23,0,10,n,0,0,0,0,0,0
```

If n=0 then the cursor will be at its biggest whilst n=10 will totally remove the cursor.

PROCEDURE INDEX

by Nicholas Sayers

The following function key definition will list all of the functions and procedures in memory along with the line numbers at which they are situated.

```
*K.0P=PA.:REP.N=P?1*256+P?
2:P=P+3:REP.P=P+1:U.?P=221
OR?P=13ORP>TOP:IF?P<>221U.
P>TOP EL.P=P+1: P.RI. "
    "+STR$(N),5);" ";M."FN
PROC",ABS(?P=242)*3,2+ABS(
?P=242)*2);:REP.P=P+1:VDU?
P:U.?P=13:P.:U.P>TOP
```

Be careful to enter the definition exactly as printed. Do not expand any of the abbreviations or Basic will not let you enter the entire string.

Once it has been entered correctly, press f0 to list all of the functions and procedures in memory. Quite a handy untility!

SIDEWAYS SCREEN

by Colin Cleaver

Screen images may be saved in sideways RAM and recalled easily on the Master and Compact micros using the *SRWRITE and *SRREAD commands. The following listing demonstrates this:

```
10 *SRDATA 4
20 *SRDATA 5
30 *LOAD SCREEN
40 *SRWRITE 3000 8000 0
50 CLS
```

Line 30 simply loads in a previously saved screen but could be replaced with the appropriate graphics commands. Line 40 writes the screen data into sideways RAM. To recall the screen, use the following:

```
60 *SRREAD 3000 8000 0
```

This will work in any mode.

Points Arising....Points Arising....Points Arising....Points Arising....

CROSSWORD EDITOR (Vol.7 No.4)

Line 2090 of the program was listed incorrectly. It should read:

```
2090 VDU24,32;32;1248;992;:GCOL0,129:CLG
```

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£ 7.50	6 months (5 issues) UK only	£23.00
£14.50	1 year (10 issues) UK, BFPO, Ch.I	£33.00
£20.00	Rest of Europe & Eire	£40.00
£25.00	Middle East	£44.00
£27.00	Americas & Africa	£48.00
£29.00	Elsewhere	

BACK ISSUE PRICES (per issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

POST AND PACKING

Please add the cost of p&p as shown opposite.

BEEBUG
Dolphin Place, Holywell Hill, St.Albans, Herts AL1 1EX
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Technical Assistant: Lance Allison
Production Assistant: Yolanda Turuelo
Membership secretary: Mandy Mileham
Editorial Consultant : Lee Calcraft
Managing Editor: Sheridan Williams

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Limited.

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

BEEBUG Ltd (c) 1988

Printed by Head Office Design (0782) 717161 ISSN - 0263 - 7561

Magazine Disc/Cassette

OCTOBER 1988 DISC/CASSETTE CONTENTS

GRAPHIC DESIGN WITH ASTAAD - a new version of BEEBUG's acclaimed graphic design program, now for the Master, and the full original version for the model B.

HOW TO BE A GOOD MOUSER - use this program to build your own mouse-controlled programs independent of any ROM-based software.

THE BEEBUG SUPER-SQUEEZE - save memory space with the ultimate compaction utility.

PRINT FORMATTING - a demonstration of a procedure which you can use to provide detailed formatting of numeric output in your own programs.

DESIGNING SCREEN AND PRINTER CHARACTERS - a test program to check your designs for screen and printer characters, supplied here with the Turkish character set from last month's article.

ELEVENSES - a very well implemented and quite addictive form of patience for a little relaxation.

FIRST COURSE

SCROLLING STRINGS - two more routines for fancy scrolling of text.

CHARACTER SORTING - an alternative approach to sorting characters within strings using a mask.

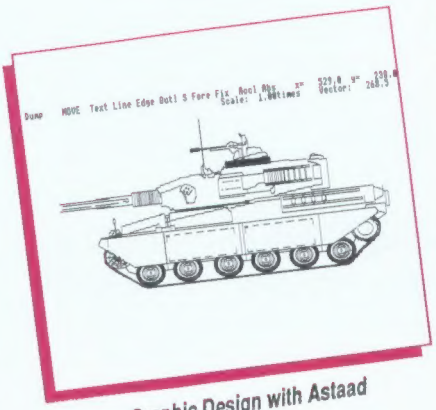
VISUAL SORTING - an implementation of three different sort methods vividly illustrated on the screen.

FILE HANDLING FOR ALL (Part 5) - a complete working database program incorporating all the routines from this and the previous two installments.

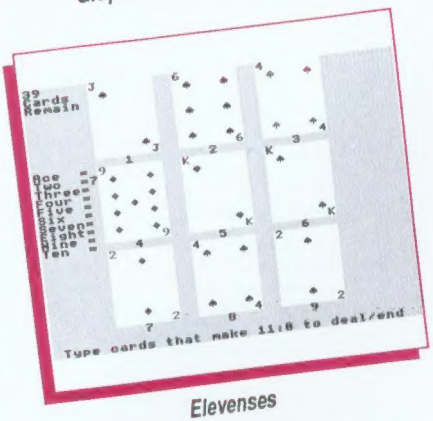
USING ASSEMBLER (Part 3) - two assembler programs showing the use of vectors, including a utility for a dual screen display.

BEEBUG WORKSHOP - two programs, one to list out all Basic's current variables, and one to demonstrate the use of heap management techniques.

MAGSCAN - bibliography for this issue (Vol.7 No.5).



Graphic Design with Astaad



Elevenses

All this for £3 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES	UK ONLY			OVERSEAS		
	5" Disc	3.5" Disc	Cassette	5" Disc	3.5" Disc	Cassette
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£39.00

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts. AL1 1EX.

Internal Master Modem

(Master-owners)...should seriously consider this excellent new product from Beebug.

...cheaper than many other modems of a similar specification...
ACORN USER OCT 88

for the Master 128

The Beebug Master Modem is a high quality internal modem for the Master 128. It is easy to install and requires no soldering. The Master Modem is supplied with the Command software which allows easy access to a whole range of services, such as Prestel, Telecom Gold, Bulletin Boards etc.

Features include

- fits internally in Master 128
- just plugs in - no soldering reqd.
- 300 baud full duplex (V21)
- 1200/75 baud half duplex (V23)
- auto-dial facility
- V25 auto-answer facility
- line monitoring through computer loudspeaker
- tinkle suppression
- supplied with Beebug's highly acclaimed Command software

Beebug Master Modem including fitting instructions, software and user guide

price £119 inc VAT

APPROVED for connection to telecommunication systems specified in the instructions for use subject to the conditions set out in them.

S/3113/3/H/501018

Certain early versions of the Master 128 had their battery packs located in the modem fitting position. If your battery pack is NOT located to the left of the keyboard, it will need replacing with the new style battery pack (price £3.68).

To claim Beebug members price it is essential to quote your membership number

Please supply ___ Master Modem(s) at £119 ❑ or £113.05 memb. price ❑ Carriage £4

Please supply ___ New style battery pack(s) at £3.68 ❑ or £3.50 memb. price ❑ Carriage £1

Name: _____ Membership No. _____

Address: _____

_____ Postcode: _____

Access/Visa No. | | | | | | | | | | | | | | | | |

ACCESS/VISA Expiry Date | | |

Beebug Ltd. Dolphin Place, Holywell Hill, St. Albans, Herts., AL1 1EX Tel (0727) 40303